BUILDING ENTERPRISE TRANSITION PLANS THROUGH THE DEVELOPMENT OF COLLAPSING DESIGN STRUCTURE MATRICES

DISSERTATION

Michael P. Kretser, Captain, USAF

AFIT-ENS-DS-15-S-033

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-DS-15-S-033

BUILDING ENTERPRISE TRANSITION PLANS THROUGH THE DEVELOPMENT OF COLLAPSING DESIGN STRUCTURE MATRICES

DISSERTATION

Presented to the Faculty

Department of Operation Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy in Logistics

Michael P. Kretser, BS, MS

Captain, USAF

September 2015

AFIT-ENS-DS-15-S-033

BUILDING ENTERPRISE TRANSITION PLANS THROUGH THE DEVELOPMENT OF COLLAPSING DESIGN STRUCTURE MATRICES

Michael P. Kretser, BS, MS

Captain, USAF

Committee Membership:

Dr. Jeffrey. A. Ogden
Chair

Dr. John M. Colombi
Member

Dr. Paul L. Hartman
Member

ADEDEJI B. BADIRU, Ph.D.
Dean, Graduate School of Engineering
and Management

Abstract

The United States Air Force (USAF), like many other large enterprises, has evolved over time, expanded its capabilities and has developed focused, yet often redundant, operational silos, functions and information systems (IS). Recent failures in enterprise integration efforts herald a need for a new method that can account for the challenges presented by decades of increases in enterprise complexity, redundancy and Operations and Maintenance (O&M) costs.

Product or system-level research has dominated the study of traditional Design Structure Matrices (DSMs) with minimal coverage on enterprise-level issues. This research proposes a new method of collapsing DSMs (C-DSMs) to illustrate and mitigate the problem of enterprise IS redundancy while developing a systems integration plan. Through the use of iterative user constraints and controls, the C-DSM method employs an algorithmic and unbiased approach that automates the creation of a systems integration plan that provides not only a roadmap for complexity reduction, but also cost estimates for milestone evaluation.

Inspired by a recent large IS integration program, an example C-DSM of 100 interrelated legacy systems was created. The solution provided by the C-DSM method indicates that if a slow path to integration is selected (10% reduction of systems per iteration), then cost savings are estimated to surpass integration and O&M costs after 3 iterations. If the constraints are held constant, the number of systems reduces to 1 enterprise system after 27 iterations and estimates nearly $4.6 billion in savings.

Dedicated to God, my family, and my country

Acknowledgments

I would like to extend my utmost thanks and appreciation to my research advisor, Dr. Jeffrey Ogden, and my committee members, Drs. Colombi and Hartman, for their guidance, support, and patience through the course of this dissertation endeavor. Their insights were instrumental in the development of my research from the earliest concept designs to the completed methods described in this document.

I am also thankful for the insightful discussions between this committee and the members of Sri-Sys who provided some application-level feedback which assisted in the development of the final evolution of the C-DSM method and metrics of interest to this field of study.

Michael P. Kretser

Table of Contents

List of Figures

List of Tables

List of Equations

BUILDING ENTERPRISE TRANSITION PLANS THROUGH THE DEVELOPMENT OF COLLAPSING DESIGN STRUCTURE MATRICES

I. Introduction

Background

The United States Air Force (USAF), like many other large enterprises, has evolved over time, expanded its capabilities and has developed focused operational silos that do not fully cross-communicate while still accomplishing the mission through the efforts of its workforce in spite of the challenges presented by unconnected systems. As enterprises expand over time through innovative evolution, or through acquiring competitors via mergers or takeovers, they become more complex. The problem is exacerbated if there is not a significant enterprise governance structure preventing business units from hiring and developing their own systems departments. “In many companies, business-IT governance is not managed cohesively or from a holistic, firm wide perspective. Instead, decisions are made in siloed fashion within individual business functions or units, with little thought given to how those decisions might affect other parts of the company or the company as a whole.” (Grebe & Danke, 2013) It is not uncommon for new additions to the enterprise to continue operating without significant change; oftentimes the only discernible indication of “integration” is the change in company logo on some of the business unit’s offices and some changes to management teams. Over time these silos become more entrenched and internally focused making it is easy for silo leadership to focus internally to strive for optimal performance utilizing internal metrics while failing to see the external affects to the enterprise at large. In the seminal article “Issues in Supply Chain Management,” (Lambert &

Cooper, 2000) Dr. Lambert and Dr. Cooper illustrate disconnects between functional silos across supply chains (SC) and this same illustration can be used to describe the silos within a large enterprise.

> "Different names were used for similar processes, and similar names for different processes. We believe that this lack of inter-company consistency is a cause for significant friction and inefficiencies in supply chains. At least with functional silos, there is generally an understanding of what functions like marketing, manufacturing, and accounting/ finance represent. If each firm identifies its own set of processes, how can these processes be linked across firms?" (Lambert & Cooper, 2000)

It is necessary to take the enterprise view of an organization to examine the holistic flow of processes from the earliest input to the final output to evaluate where change is needed to reduce costs, reduce waste, and improve the flow of information between subordinate organizations. Given the complexity of an enterprise this is no easy task and to accomplish this one must first develop a model of what an enterprise is and employ methods to understand the model. From General Systems Theory (GST) we can view the enterprise as a large system composed of sub-systems and utilize systems engineering methods to analyze and harness the complexity of the system model. Systems engineering has expanded to system-of-systems engineering (SoSE) and more recently enterprise SoS has surfaced to specifically focus on enterprise complexity in a systems view (Rebovich Jr., 2008:165-189). As organizations become larger it becomes more challenging to understand the complexity of the connections between sub-systems as some systems touch many systems while others only provide minimal information to other sub-systems. When the number of systems is in the

hundreds each with tens to hundreds of connections it can be a daunting task to map and fathom the entirety of an enterprise. When the enterprise is experiencing a crisis such as bankruptcy and needs to look internally to eliminate waste and integrate processes then they need to utilize some tools and methods to sift through the tightly woven highly complex web of sub-systems. Many enterprises have experienced these trials and some have successfully survived by making hard choices and employing process improvement techniques such as Business Process Reengineering (BPR), Total Quality Management (TQM), Lean, Six-Sigma, or implementing Enterprise Resource Planning (ERP) software to connect systems and realize some business process improvements. Many have tried several of these techniques and have experienced utter failure; others have implemented an ERP successfully while not experiencing the expected Return on Investment (ROI) (Umble, 2003).

An enlightening paper by (Fu Xiang-ling, Song Mao-qiang, Yu Ya-nan, & Mian, 2010) describes the root of these problems and contains some effective analogies. Over a century of task separation, industrial process improvement, and departmentalization combined with the birth of the computers and automation resulted in "islands of information" that did not cross-communicate. These "islands" were not designed for future integration when the information age took root. Within these islands of information separate systems were collecting similar data (frequently with inconsistency in data type, naming convention, etc.), but could not share the data resulting in the need to build bridges between these islands. They offer a few ways to approach this problem. The first is to completely start fresh with a new unified data system if resources are available and there is a solution that meets the needs of the organization. The second is to build a unified "nervous system" to integrate the

islands, such as middleware supported by software "wrappers" that can communicate with a new primary system. The third way to address these disjointed systems is integrate groupings of systems that may cross departments (Fu Xiang-ling et al., 2010). It comes down to how complex the enterprise problem is, the amount of resources available, and how willing the enterprise is to accomplish any one of the three ways.

Another option companies are taking in this era of cloud computing is converting to a shared services model where legacy systems are attaching to a common middleware system (a.k.a. an abstraction layer, backplane, integration system, etc.) and broadcasting their services to those who can use them. Shared services have shown promise as legacy systems do not have to be replaced, but upgraded and connected through middleware presenting a less costly solution. However, without first reducing the number of systems in use, the number of shared services will exponentially increase the amount of complexity as architecture to support the shared services adds systems and database overhead to the enterprise without removing systems.

Problem Statement

The path to developing and implementing IT systems integration eludes executives as many choices and decisions must be made without any tools to support their decisions. "Companies require a realistic route to implementation that sequences migration and places the services within a coherent organization design that ensures rigorous, effective governance." They typically do not know what the end state should look like or where to begin (Roghe, Toma, Messenbock, Kempf, & Marchingo, 2013). Companies need an analytical tool that will provide an unbiased, systematic, and cost-effective roadmap. Bias is

minimized through use of an algorithmically generated solution. The algorithm is systematic in nature meaning that the decision criteria remain the same for cluster identification. Cost effectiveness is defined as the minimization of the financial burden of the solution provided. The C-DSM method described in this research has an objective to minimize complexity by reducing systems to provide savings through reduced maintenance costs.

Research Objective and Questions

The overarching objective of this research is to develop an algorithm-based method that can optimally integrate a large number of enterprise IT systems over time. Through the course of the research several research questions were also answered that were directly beneficial to the development of a practitioner-focused application:

- What are other methods for identifying and removing redundancy (and therefore complexity) in large firms?
- What are the costs (money, time, other resources, etc.) of (or refraining from) integrating a large complex enterprise?
- How should firms/enterprises evaluate systems prior to integration? What are some valid taxonomies for different business types (e.g. retail, logistics, military, etc.)?
- What are the pros and cons of complete integration versus a spiral development approach to integration?

Research Focus

Many enterprises today are not fully aware of how many redundant systems they employ or to the extent of the redundancy. Other enterprises are aware of the complexity of the interwoven dependencies of systems but are unsure on how to begin to unravel the web of complexity. Still others have invested in ERP software to integrate their firms or have invested much time and effort into transforming their businesses using TQM, Lean, BPR and

other process improvement initiatives with varying levels of success and return on investment. However, there does not appear to be a method on how to employ these initiatives other than through committee or project team formations of experts. A systematic tool for evaluating all systems in an enterprise and proposing a logical plan for integration does not exist in academic literature. Without this method, experts are left to make "gut decisions" without the benefit of an academically supported method. One method for reigning in complexity suggested by the Boston Consulting Group involves "a multipronged approach," but lacks the details on how to implement the approach. Potential clients are referred to paid consultants. . The six approaches are: intelligent demand management, scenario-based application rationalization, infrastructure technology-pattern reduction, a simplified IT organization and an enabled IT workforce, effective governance and simplified processes, and finally, a shared-services model and optimized sourcing (Grebe & Danke, 2013).

To implement and sustain systems integration a few ideas can be gleaned from another IT centered study by BCG where they detail four "must-haves" which coincide with other authors cited in this study. The first of the four "must-haves" is "a blueprint of the target end state and a roadmap for getting there." This runs parallel with the need for the "TO-BE" as noted by systems engineering authors (Dagli et al., 2013; Inspector General of the United States Department of Defense, 2012; Riposo, Weichenberg, Duran, & Fox, 2013) and the roadmap that this dissertation is attempting to create. The second "must-have" is "a program management office helps drive progress and track results." This relates to the need for an enterprise integrator with methods of tracking the progress at an enterprise view

(Ahmad & Cuenca, 2013; Dagli et al., 2013). The third "must-have" is to set enterprise policy to solidify change and have a governance structure to ensure compliance and conflict resolution which coincides with research presented by (Riposo et al., 2013; Strachan, 2008). The final "must-have" is "buy-in from senior business management (the CEO and CFO) and senior IT management (the COO and CIO)." This is repeatedly listed as a critical success factor in numerous studies such as (Hammer & Champy, 2003; Umble, 2003; Wang, Shih, Jiang, & Klein, 2008).

In addition to these four "must-haves" change management is touted by each of these authors as one of the most crucial factors for any process improvement effort. Table 1 summarizes these ideas and will be referred to as the pillars of systems integration implementation and sustainment. These pillars could apply to any process improvement effort, but is more appropriately applied when large complex organizations such as enterprises are concerned where big ideas at the top of a distributed hierarchy is more apt to morph considerably as they trickle across and down the organization where the governance structure and integration office is most important. Figure 1 illustrates the general hypothesized relationship of these factors and the highlighted focus factor under study in this research.

Table 1. Pillars of System Integration Implementation and Sustainment

Pillars of System Integration Implementation and Sustainment	**Supporting References**
"TO-BE" Target and Roadmap	(Dagli et al., 2013; Inspector General of the United States Department of Defense, 2012; Riposo et al., 2013)
Enterprise Integration Office	(Ahmad & Cuenca, 2013; Dagli et al., 2013)
Governance Structure	(Riposo et al., 2013; Strachan, 2008)
Top Management "Buy-in", Support, and Championing	(Hammer & Champy, 2003; Umble, 2003; Wang et al., 2008)

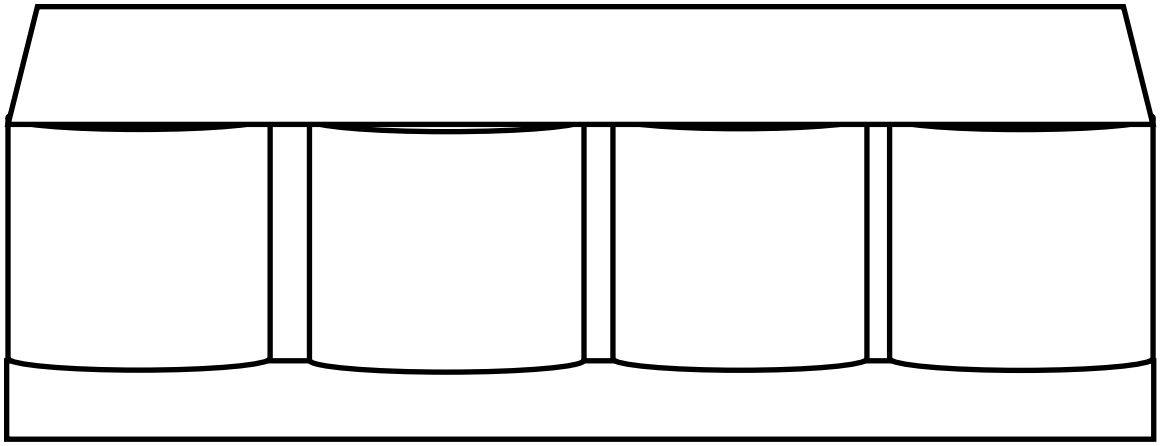


Figure 1. Pillars of Systems Integration Implementation and Sustainment

Methodology

This research developed a mathematical algorithm that produces transition plans, or roadmaps, for the reduction of complexity of an enterprise systems network using a novel application of DSM concepts. This method displays the "As-Is" state of enterprise systems and provides a stepwise progression to obtain a logical "To-Be" architecture for the purpose of strategic enterprise decision making without the biased influence of silo-protecting politics that are frequently present in hierarchal enterprises and large firms (Lambert & Cooper, 2000; Motwani, Subramanian, & Gopalakrishna, 2005; Roghe et al., 2013; Umble, 2003; Wang et al., 2008). Complexity should diminish as systems are integrated or removed from the enterprise while the functions provided by these systems are preserved by the remaining systems or divested as superfluous functions that are not aligned with enterprise goals.

Assumptions

This research introduces a collapsing DSM (C-DSM) for the purpose of planning systems integration. Based on prior research in networking and complexity theories, it is assumed that through the reduction in the number of systems, complexity will naturally diminish and the relationships between the fewer but larger systems will become clearer. Other techniques to reduce complexity, or waste in the business sense (TQM, BPR, Lean, Six-Sigma, etc.), describe the needs for simplification and identify the presence of wasteful excess in firms but do not describe in detail the actual steps to execute these techniques, but instead speak in the more general sense. This research should be beneficial as it complements these broader techniques with a specific tool for developing transition plans.

Limitations

As with any model or method there are risks involved as George E.P. Box famously stated, “All models are wrong, but some are useful.” The algorithm developed will only be as effective as the factors employed in the calculation or in the amount of detail provided in the source matrices. As with any model or method, if the wrong factors are employed to the wrong business model then a less than optimal solution (or a detrimental solution) will result (the “garbage in, garbage out” axiom). However, if the right factors are chosen to represent the problem modeled, then a logical solution should result.

Implications

Any large corporation or enterprise that is vast enough to be divided into divisions and follows a hierarchal business model should benefit from the methods in this research. Large private sector corporations such as IBM and Wal-Mart, as well as large government

departments such as the Department of Defense and the Department of Homeland Security could benefit from this research (U.S. Government Accountability Office, 2008; Umble, 2003)as the primary focus is reducing highly complex organizations that experience a broad range of stakeholders and follow complex (and possibly overlapping) bureaucratic policies.

If successful, these methods could provide CEOs and/or action officers with a non-political decision making tool to map, score, and recommend system integration and/or cancellation. Reduced operating costs will result from the streamlined enterprise resulting in a more optimal organization increasing profits in the private sector and a gold mine of cost savings for government spending once the cost of integration is recovered.

Preview

In Chapter II of this dissertation, a literature review of the problem presented in Chapter I is conducted. In this review, the problem is examined and supporting theories are offered to provide context and construct a basis from which to develop a relevant integration tool. Chapter III is a paper submitted to the *Journal of Enterprise Integration*. In this review, a detailed literature review on DSMs is conducted and identifies a lack of DSM use for resolving enterprise IT challenges. Chapter IV is an article that has been submitted to the journal *Systems Engineering* and articulates the C-DSM methodology developed through the course of this dissertation research. Next, Chapter V is a paper near completion that will ultimately be submitted to the journal *IEEE Transactions on Engineering Management* where a case study is utilized to illustrate the C-DSM method. The case study represents an actual government legacy system integration effort currently in progress and provides and illustrative example on how to construct the required matrices, determine user constraints,

and initiate the subsequent transition plan. Finally, Chapter VI provides conclusions, and significance of this dissertation; provides recommendations for action; and elucidates several avenues for future research.

II. Literature Review

Chapter Overview

The purpose of this chapter is to introduce the supporting theories and mathematical methods that were eventually used to develop a new method in the field of DSM research. This chapter provides more detail than is presented in the format of the proceeding chapters, as the proceeding chapters are formatted and reduced for publication in peer-reviewed journals. This chapter begins with the most general and progresses to the more focused of the theories that support this research. Following the review of systems theories which form the basis of this research, a review of the foundational and historical DSM methods is presented. This chapter concludes by directing the reader to the following chapter where a focused and methodical literature review is conducted.

Relevant Research

Systems and Networking Theories

As networking theory is a subset of systems theory it is necessary to first examine the basics of systems (and SoS) theory as it will provide a basis for the rest of the following discussions. Systems theory in its most general sense is the study of systems and how their subsystems are organized and relate to one another. Various fields of research have expanded this theory and applied it accordingly, but originally it was viewed in the organic or natural sense. The term ‘systems theory’ originated from Ludwig von Bertalanffy in the 1950s and he was known for championing the thought of open systems that reacted to their environment

as he initially studied humans and their systems in reaction with the environment (Von Bertalanffy, 1951; Von Bertalanffy, 2008)(Von Bertalanffy, 1972). As a system is composed of yet more systems it becomes a matter of where one wants to scope their investigation to analyze behaviors referred to as a system-of-systems (SoS).

Systems theory is a generalized theory of viewing complex groupings of systems as they relate to each other in coordination for a common goal. An enterprise is a complex SoS that must work together to reach a common goal in reaction to its environment (Ponomarov & Holcomb, 2009). General systems theory applies to enterprises as they can and will react to outside stimulus as if enterprises were single organisms or systems.

Now that systems (or SoS) can be viewed as entities (a.k.a. systems, agents, elements, and others) the need to understand the connectivity between these entities can be accomplished using network theory. Network theory is the study of graphs of objects and their relationship or connection to one another (Su-Yu Liu, Li, Yi-Ping Feng, & Rong, 2013). A graphical depiction of the logical framework of how a series of distinct entities "nodes" are connected through lines "or arcs" make up a network. Current network theory is the product of graph theory from mathematics dating back to Leonhard Euler in 1735 (Alexanderson, 2006) and early network theory from social sciences in the 1930s which were brought together by Cartwright and Harary in 1956 (H. C. Lee, July, 2009). Creating network graphs allows us to analyze complex problems without the complication of including all of the physical details, but reducing the complete problem to this simplest form of nodes and paths (H. C. Lee, July, 2009).

Viewing an enterprise network as a single organism which decomposes to a SoS that works together in response to its environment can be further understood by first mapping the connections between the systems using networking theory and quantitative methods previously used in the networking domain. As the overarching focus of my research is to map and cluster an enterprise's systems network it is important to first understand what systems are connected in a network setting so it can further be reorganized through clustering methods to make sense of the vastness of the enterprise and construct a quantitative method to formulate strategy recommendations for systems consolidation. As stated by Bezuidenhout et al. (Bezuidenhout, 2012), "[Network analysis] provides an invaluable tool for systematically assessing a network and identifying critical points within the network where interventions can be targeted." In the adoption of systems and networking theories the DSM method to be employed in my research has a theoretical basis for operation.

These theories form the basis for the applied context of viewing enterprise IT systems as a network or system of systems that can be examined as a collected whole for the purpose of consolidating redundancy and sharing information. These theories seem to be appropriate in the formulation of a DSM and this dissertation hypothesizes that applying clustering algorithms will provide a strategic map for decision makers to apply previously proven integration techniques. A clustered DSM should provide a basis for producing multiple SoSs that can be connected on a single communication layer until eventual integration. A single SoS connected via a backplane was proven by the DoD in the creation of TBMCS (Collins & Krause, 2005) and this research proposes that this can be done repeatedly in sizeable portions that can be connected via a single backplane after each SoS is created.

The contribution of this research is to refine these theories in providing a modified method in a new context. Supply chains can be extremely convoluted and complex and are usually examined at the dyadic level (Pathak, 2007) where a focal firm is related to one or two tiers up and down the supply chain whereas the use of a DSM would take a systematic purposeful approach to fully map the supply chain as far as possible. In large enterprises such as the DoD, functions and departments are so vast that the details of how they connect can easily get lost to decision makers. Turning an inward examination of complex entities such as this and taking the time to map them should allow for connectivity improvements, and eventually firm and supply chain performance (Sanders, Autry, & Gligor, 2011). With current technology, and examples of standards integration such as internet protocols, it should be possible to map and integrate legacy IT systems once their redundancy and need for integration is identified. As stated by Lambert and Cooper, "A prerequisite for successful SCM is to coordinate activities within the firm" and their SCM integration model includes information flow across the SC to include within the focal firm (Lambert & Cooper, 2000).

Complexity Theory

The next theory in this research is complexity theory which is derived from systems theory. To explain complexity theory Honour and Browning (Honour & Browning, 2007) first describe the spectrum between order and chaos. Systems engineering is an orderly perspective where everything is put in a place and the systematic placement or classification of elements brings order out of the unorganized. Chaos is the opposite of order where everything is wild and has free range to operate without rules. However, Honour and Browning note that even chaos when viewed from a high enough vantage point will

eventually show some predictability if provided mathematical and statistical tools. Complexity is somewhere between order and chaos in that systems (entities) that are placed (or have a place) in a system (or environment) will continue to operate and behave independently. Chaos allows entities to be creative and evolve (or improve themselves) while order provides a unified direction or understood ground rules that are provided by the environment. Containing both order and chaos allows complex systems to adapt to change when external forces come to bear. "[C]omplexity [theory] often concerns non-linear relationships between constantly changing entities. Systems theory, in contrast, studies static entities linked by linear relationships…complexity research concerns how complex behavior evolves or emerges from relatively simple local interactions between system components over time." (Manson, 2001)

In a SoS context, these entities (or systems) are viewed as agents as they make up the smallest individual system in an environment that work together with the other systems in the population to work towards a common goal. The SoS provides a unifying structure to bring systems together that otherwise would not come together. The independent nature of agents is important to complexity theory because it is through agent behavior with the environment and each other (termed reflexivity) that produces emergent behavior. Agents make decisions on how to respond to influences based on local information, meaning they only have knowledge about what is happening in their local scope of understanding and are for the most part oblivious to the world around them. When looking at a SoS like a large enterprise this corresponds to the silos mentioned in the introduction of this research. Managers will lead their organizations as best they can in relation to the knowledge they have. If understanding

outside of their silo is not provided they will sub-optimize in an effort to increase their performance based on their own perceptions and metrics. Honour and Browning continue to explain emergent properties or behaviors as those that "are perceptible only at the system level and cannot be perceived or even predicted from the behaviors of the parts (Honour & Browning, 2007)."

Termite mounds are frequently illustrated when discussing these theories to describe self-organization as their construction is determined by a plethora of agents (termites) who act independently based on their internal programming of simple instructions and local information yet construct massive structures without any centralized control (Honour & Browning, 2007)(Kauffman, 1995).

A final element of complexity theory that must be mentioned is that of complex adaptive systems (CAS). This element explains the adaptability of a system to improve itself in relation to the environment and its structure for the greater good of the structure (the SoS). "A CAS is an interconnected network of multiple entities (or agents) that exhibit adaptive action in response to changes in both the environment and the system of entities itself (Choi, Wu, Ellram, & Koka, 2002). Collective system performance or behavior emerges as a nonlinear and dynamic function of the large number of activities made in parallel by interacting entities. For example, the individual decisions made by firms facing imperfect information and variable demand lead to a globally observed phenomenon (i.e., the bullwhip effect) (Lee, Padmanabhan, & Whang, 1997)." (Pathak, 2007) Honour and Browning cite (Waldrop, 1992) for creating four mechanisms that produce a CAS: positive feedback, negative feedback, balance of exploration and exploitation, and multiple interactions. "When

all four of these mechanisms exist, then the self-organization of a complex system moves into the realm of CASs. These characteristics of CASs are of most interest to us, because a SoS without central control develops through these mechanisms." (Honour & Browning, 2007) The evolution of CAS is constantly in motion over time as each agent is constantly moving, adapting, and influencing each other making static mathematical approaches to understanding CAS challenging.

The study of complexity theory is relevant to this dissertation research as enterprises are SoS that generally have evolved over time. Large enterprises of our times are conglomerations of what used to be many other companies that evolved over time under their own direction and knowledge. One does not have to search long on the internet to find that a few large organizations now own the vast majority of products at the supermarket; enterprises such as Proctor and Gamble, Johnson and Johnson, Kraft, Coca-Cola, Pepsico, and others have accumulated other companies through mergers and takeovers creating vastly complex SoS. It is when these enterprises try to connect legacy systems of previously disparate organizations that have nothing technically in common yet provide the same functionality and services that this research attempts to understand and elucidate for SoS managers. In an AF example, AF logistics IT has grown over time as organizations within the enterprise developed computer aided systems before the information and networking age based on their own local information. These AF legacy systems provide similar functions for their silos but do not necessarily communicate with others in the SoS or with enterprise-level management. These CAS must be mapped to provide enterprise-level understanding for the purpose of consolidation and integration, bringing order from complexity.

One pivotal article that helped grow, direct, and support this research is presented by Madni and Moini (Madni & Moini, 2007). In this article the authors apply "systems thinking to enterprise transformation, management, and evolution" through exploration of "systems, complexity, autopoiesis, social network, complex adaptive systems, and net centricity theories." This conceptual paper first lists the characteristics of a SoS as presented by (Maier, 1998): operational independence of the elements, managerial independence of the elements, evolutionary development, emergent behavior, and geographic distribution. The authors point out that the "focus of the modern enterprise is creation of sustainable value" while "in a SoS the primary focus is creation of sustainable capability." The authors suggest that an enterprise SoS explicit goal is therefore "to cost-effectively create, sustain and evolve a capability."

The authors then list the characteristics of the modern enterprise which holds much similarity to the characteristics of a SoS. They are: Agility (ability to rapidly change), Emergence ("large-scale, aggregate behaviors"), Cross-Functionality ("seamless integration of multiple business domains"), Decentralized Control ("loose coupling among organizational units" through "relegating authority"), Dynamic Membership (form and disband "temporary alliances and integration with other systems"), Expandability, Hierarchical Structure ("most enterprises are complex, dynamic webs of interaction"), Scalability, Self-Organization, Structural Determinism (reorganize to respond to stimulus), Structural Openness (allows "late arriving stimuli/information/material from the environment" and reorganize accordingly), Superconnectivity ("information diffusion both at the micro and macro level"), and Virtuality ("temporary arrangement to respond to a specific business need" which is related to emergence and agility) (Madni & Moini, 2007).

The authors (Madni & Moini, 2007) then explore enterprises and autopoiesis theory which explains self-organization. Attributes of autopoiesis are: Self-creation, Self-configuration, Self-regulation, Self-steering, Self-maintenance, and Self-reference. The authors state, "[t]he key aspect of autopoietic theory that is of greatest interest to understanding enterprises is its application to the study of human interactions and the social settings in which they occur." They further argue that "enterprises are autopoietic systems in that they generate, regulate, renew, repair, or regenerate their organization through the network of interactions that characterize them as well as through the flow of energy or information they send or receive from their environment." (Madni & Moini, 2007)

This paper supports using a SoS-based approach to enterprise IT architecture and data environments as it provides the theoretical groundwork for making the enterprise SoS claim and allows for the continuation from the organizational structure to the IT structure.

The contribution of this dissertation research to the theory of complexity is in the application of an evolving method (DSM) and applying it to an enterprise SoS. This has not been done before and aids the refinement of these theories by providing a case study of a conceptual application. Other methods are currently employed to harness complexity as explained in *Systems Engineering Tools and Methods* (Kamrani & Azimi, 2011) where the following methods are explained: functional decomposition (hierarchy diagrams), object oriented descriptions (UML, SySML), flow charts, functional flow diagrams, functional interface analysis (N^2 diagrams), timeline diagrams, interface definition (input, output, signals, etc), and dependency matrices (Pineda & Smith, 2011, p.35-79). DSMs continue where N^2 diagrams end as N^2 diagrams are flat. A DSM can take an N^2 diagram and perform

clustering algorithms to provide insight that is not easily seen in a large flat disorganized matrix. As N^2 diagrams are frequently used by the DoD (via DoDAF) to describe the "As-is" state of a system (and possibly an SoS) it is possible to translate available N^2 diagrams into DSMs for analysis in addition to creating DSMs from other enterprise documentation.

Design Structure Matrices

Design Structure Matrix methods are used in many ways to decompose complex systems or systems-of-systems for in depth study and analysis to understand how these complex systems relate to one another (Browning, 2001a). A matrix is created where all of the systems are listed across the columns and down the rows to create a square matrix where the diagonal element is a system mapped to itself (e.g. the first row and the first column are the same system). There are many applications for DSM and many specialties have employed their own approach on the concept such as Deng et al. in their use of DSM to separate outsourcing components to ensure protection of intellectual property rights (Deng, Huet, Tan, & Fortin, 2012), Farid and McFarlane who used DSM to assess reconfiguring distributed manufacturing systems (Farid & McFarlane, 2006a), and Cai, Iannuzzi, and Wong who used DSM to evaluate software design student homework submissions (Cai, Iannuzzi, & Wong, 2011). Batallas and Yassine combined DSM and social networking analysis to identify key communicators in a large aircraft engine company which allowed that company to invest in key people to create a new communication team as those members were the most connected members in the company with those members coming from separate work centers or clusters as displayed in the DSM (Batallas & Yassine, 2006).

The most basic DSM described by Browning maps the system behavior on the row as a relationship where it provides something to the other systems and a "1" is placed on the systems that row feeds or provides information. When looking at the systems as depicted on the columns entry, those systems depend on the rows below to feed the column entry. If all connectivity between the systems is two-way communication (e.g. A provides a benefit to B, B provides a benefit to A) then a symmetrical matrix is the result (Browning, 2001a).

Beyond the most basic DSM there are more complex uses and methodologies for employing DSMs and Browning explains various techniques tailored to the type of problem to be analyzed (see Table 2). When timing is critical to the structure of the process or problem under analysis the time-based DSM type requires analysis through sequencing while clustering is employed to analyze static problems where the relationship is more important than the timing.

Table 2. DSM Types and Analysis Methods (Browning, 2001)

	DSM Type	**Representation**	**Applications**	**Integration Analysis via**
Static	Component-Based or Architecture DSM	Components in a product architecture and their relationships	System architecting, engineering, design, etc.	Clustering
	Team-Based or Organization DSM	Individuals, groups, or teams in an organization and their relationships	Organization design, interface management, application of appropriate integrative mechanisms	
Time-based	Activity-Based or Schedule DSM	Activities in a process and their inputs and outputs	Project scheduling, activity sequencing, cycle time reduction, risk reduction, etc.	Sequencing
	Parameter-Based DSM	Parameters to determine a design and their relationships	Low-level process sequencing and integration	

Once the DSM is mapped, the analysis must be accomplished to produce insights into the amount of integration or connectivity in the program, process, or network under study.

Clustering is the reorganization of the DSM which is meant to minimize distance or maximize interactions of the relationships between the components or systems. The systems will normally shuffle to create groupings (or clusters) of highly connected systems and minimize the distance between end-to-end communication of systems that must share information. What type of algorithm employed to accomplish this shuffling depends on the type of problem under study and the type of relationship to be analyzed. The example Browning uses to explain one method for clustering comes from a DSM made for Ford Motor Company where the components of the climate control system in an engine were mapped to show how much different components depended on each other. After seeing the clustered DSM Ford was able to propose changes to component layout as well as explore the idea of combining engineering design teams that may not have interacted much previously.

Further discussion of the body of knowledge for this research is covered in a literature review paper that has been submitted to a peer-reviewed journal for publication in Chapter III.

US Air Force Integration Shortfalls

After years of struggle with implementing ERPs the AF commissioned the RAND Corporation to research "key early planning issues associated with ERP programs" along with recommendations for the USAF and "how these key early planning issues may be manifested during program execution" also with recommendations for the USAF to improve early program assessments. This report was conducted from 2011 to 2012, but was published in 2013 after the cancellation of the Expeditionary Combat Support System (ECSS). In this RAND report "Improving Air Force Enterprise Resource Planning-Enabled Business

Transformation" (Riposo et al., 2013) the authors investigate the conditions for successful ERP implementation and break these down to: the Business Case, Governance, Business Process Reengineering (BPR), Organization Change Management, and IT Acquisition. For each of these conditions they provide challenges that the USAF will have to overcome to be successful.

The RAND report states that the Business Case is used to justify whether or not a project should be undertaken and is used as a tool for subsequent planning and management. If done correctly it should "articulate the transformational goals and desired benefits that are aligned with an enterprise business strategy." It should be the "framework for cross-functional decision making" and the point at which functional areas connect. The Business Case is contingent upon a clear current "AS-IS" environment and the target "TO-BE" environment which achieves enterprise goals to include cost and performance (Riposo et al., 2013). The authors also conclude that the USAF has struggled with enterprise business strategy and creating the "AS-IS" and "TO-BE" environments.

The "Governance" factor is the decision making method that is employed to achieve a corporation's enterprise-focused goals. To employ this correctly the authors recommend a single responsible authority or small group. However, the USAF is not aligned in this manner as far as a department-wide strategy or in an organizational structure. The USAF, and most likely other military departments, has an elusive organizational structure with competing objectives and unclear seniority. In the HQAF organizational chart the dual structure of operations vs. business functions can be seen. Not only are decision-makers in their positions for a short period of time, they are horizontally aligned, given equal power, but responsible to

different objectives while potentially overseeing stove-piped functions which can lead to sub-optimization at the expense of enterprise goal or performance achievement. As the authors so eloquently state, "A consequence of a multilayered, cross-functional governance structure is that information can be filtered, diluted, or otherwise changed. This potentially delays critical decisions or even results in the wrong problem being addressed." (Riposo et al., 2013) As if the challenges to the USAF were not enough the problem is further complicated the USAF's responsibility to operate in accordance to a web of federal laws, regulations, and policies that currently contradict each other on some level that will have to be updated or supplemented to create a consolidated ruling governance structure (Riposo et al., 2013).

Mitigating USAF Shortfalls

To attempt to mitigate some of the shortfalls identified in the RAND report, the methods presented in this research attempt to methodically drive integration separate from organizational structure, and frequent leadership change by mapping relationships across the enterprise. The methods allow users to model adherence to law and policy change timelines through the creation of weighted relationship matrices and constraints. Once enacted, the model will produce transition plans that can be moderately modified (or left alone) by the current leadership, but will continue long after the leadership change has occurred; planning is no longer dependent on one or two key individuals. Developing the source matrices provides the current "AS-IS" enterprise structure and the output of the algorithm will provide iterations of decreasing size and complexity of "TO-BE" states of the enterprise.

Summary

In this chapter, systems theories that are related to the problem presented in Chapter I were reviewed for applicability. General systems theory was first introduced and describes how an individual system reacts to its environment. Then, networking theory was reviewed and found to allow for the study of populations of systems and methods for simplifying complexity for the allowance of studying higher-level problems. Complexity theory was then reviewed and provides the basis for this dissertation research as an enterprise is a SoS that is a collection of systems that react to their environment, but work together towards the common good of the enterprise.

This chapter continued to discuss some graphical systems engineering management methods and found that DSMs were best suited for this dissertation. Basic DSM types and applications were then reviewed followed by a few examples of recent research where DSM methods were applied in different contexts for varied problem sets.

Chapter II concludes with discussing USAF shortfalls found by third party agencies that identify and further explain the problem presented in Chapter I. The research described in Chapters III-VI will provide a mitigation option to fill several of the gaps identified by the RAND Corporation that are relevant to the theories described in this chapter.

III. Exploring Design Structure Matrices to Reduce Enterprise Information Systems Complexity

Complexity and redundancy in large enterprises have grown over the last few decades and have not yet been extensively studied using optimal automated algorithms. Design structure matrices (DSMs) have been shown to be useful for analysing and clustering complex interacting components. However, the preponderance of DSM literature has focused on product or system-level problems with little focus on enterprise-level issues. This paper introduces the problem of enterprise information system (IS) redundancy and conducts a literature review of methods for developing roadmap generating algorithms to iteratively reduce that redundancy. The literature supports the proposition that DSMs provide a suitable technique for this purpose, which is highly relevant when evolving many legacy IS toward an integrated Enterprise Resource Planning (ERP) solution.

Keywords: System analysis and design; enterprise architecture; design structure matrices (DSM); information systems; systems integration

Introduction

Many large enterprises find themselves in a difficult operating environment under strict budget constraints. Often, enterprise-wide transformation is required to re-engineer a company's processes, products, services and divest itself of wasted redundancy (Gerstner, 2002; Lefever, Pesanello, Fraser, & Taurman, 2011). One challenge is how to best develop a transition plan toward the future "to-be" architecture through IS integration and shutdown. The complexity of the problem is due in part by the evolution of business, as processes were broken down into functional silos in the industrial revolution, and then optimized through

information technology and large software systems. Silos were unaware of each other's continual upgrades, and thus competing organizations developed proprietary systems to accomplish similar goals. It is only natural that these enterprises have evolved into a complex network of disconnected and managerially separate information systems. Large government enterprises such as the Department of Defense (DoD) and the Department of Homeland Security (DHS) have also evolved over a long history of separate successful organizations that have merged into monolithic conglomerates (U.S. Government Accountability Office, 2013). For example, the Government Accountability Office (GAO) reported in 2007 that the DHS still had over 500 financial systems that have been identified for integration into a single system (U.S. Government Accountability Office, 2007). Another example of this enterprise environment was the highly publicized recent failure of the U.S. Air Force Expeditionary Combat Support System (ECSS), which was trying to replace approximately 250 core logistics systems with a single Enterprise Resource Planning (ERP) solution (Reilly).

From general systems theory, the enterprise can be viewed as a large hierarchical system composed of highly interconnected sub-systems. Systems engineering has expanded to consider system-of-systems engineering (SoSE) and more recently, "enterprise SoS" (ESoS) (165Rebovich Jr., 2008189). As organizations expand, it becomes more challenging to understand the totality of the relationships between sub-systems, which have varying degrees of connectedness. When the number of subsystems is in the hundreds, each with many connections, it can be a daunting task for systems engineers and chief executive officers (CEOs) to understand the entirety of their enterprise. When the enterprise

experiences an economic crisis, and needs to look internally to eliminate waste and integrate processes, system analytics will be required to sift through the tightly woven highly complex web of subsystems. Many enterprises have been successful by employing techniques such as Business Process Reengineering (BPR), Total Quality Management (TQM), Lean, Six-Sigma, or implementing an ERP. Many have tried several of these techniques and have experienced utter failure, while others have implemented an ERP successfully, but have not experienced the expected return on investment (ROI) (Umble, 2003).

Fu et al. describes the root cause of these problems and suggests some effective solutions (Fu Xiang-ling et al., 2010) to address these "islands of information." Within these islands of information, separate systems were collecting similar data (frequently with inconsistent data types, naming conventions, etc.), but could not share the data efficiently resulting in the need to build translators between these islands. A few ways to approach this problem include: starting with a new unified data system, build a unified "nervous system" to integrate the islands, or integrate groupings of systems that may cross departments (Fu Xiang-ling et al., 2010).

Another option companies are taking in this era of cloud computing is converting to a shared services model where legacy systems are attaching to a common middleware system, an abstraction layer, and broadcasting their services to those who can use them. However, without first reducing the number of systems, the number of shared services will continue to increase complexity. The path to implementing a shared services model also eludes executives as many choices and decisions must be made without any tools to support their decisions. "Companies require a realistic route to implementation that sequences migration

and places the services within a coherent organization design that ensures rigorous, effective governance." (Roghe et al., 2013) Companies typically do not know what the end state should look like or where to begin.

Enterprise architects need tools and methods that provide a roadmap that allows them to see their current state (the "As-Is"), their desired state (the "To-Be"), and a transition plan. There are standard representations (either with or without methods) for developing the As-Is and the To-Be architectures, supported by various architecture frameworks, such as the Zachman framework (Zachman, 2010), the Open Group Architecture Framework (TOGAF), the Department of Defense Architecture Framework (DoDAF), and several others. However, there is little in the literature explaining how transition planning should be accomplished. Currently, many of the techniques employ expert opinion and "gut" decisions by process owners leading to varying results. The literature review will support the proposition that DSM methods are suitable for developing roadmap-generating algorithms to reduce redundancy and complexity of enterprise information systems.

System Theory and Systems Tools

Systems Theory

The term "systems theory" originated from Ludwig von Bertalanffy in the 1950s and he is known for championing the concept of open systems that react to their environment (Von Bertalanffy, 1951; Von Bertalanffy, 1972; Von Bertalanffy, 2008). Every organization and information system in the enterprise, as well as the entire enterprise itself, has an external context – its environment. Drawing from Bertalanffy, the organization's information

systems react to this dynamic, external stimulus in complex ways (Ponomarov & Holcomb, 2009). When studying the growth of enterprises, organizations and information systems of the enterprise behave in an evolutionary and emergent manner.

The Chief Information Officer (CIO) Council defines an enterprise as "an organization supporting a defined business scope and mission. An enterprise includes interdependent resources (people, organizations, and technology) that must coordinate their functions and share information in support of a common mission (or set of related missions)." (Chief Information Officer Council, 2001) From an evolutionary standpoint, most large enterprises are the aggregate of previously separate corporations (and their IT systems).

One can view the organizations and information systems in the enterprise as agents. An agent is an abstraction of the smallest individual system in an environment that exhibits (independent) autonomous behavior. The independent nature of agents is important to complexity theory because it is through agent reaction with the environment and each other (termed reflexivity) that produces macro-level emergent behavior. Agents make decisions on how to respond to external influences based on local information. Local decisions often result in sub-optimized enterprise performance. Honour and Browning explain emergent properties or behaviors as those that "are perceptible only at the system level and cannot be perceived or even predicted from the behaviors of the parts" (Honour & Browning, 2007) (Kauffman, 1995).

Honour and Browning state that in order to explain complexity theory, it is important to describe the spectrum between order and chaos (Honour & Browning, 2007). Order is deterministic behavior and systems engineering can be considered a disciplined process to create order (man-made solutions), from a set of unordered changing set of requirements, stakeholders and components. Chaos is the opposite of order where system behavior and evolution appears random over time. However, Honour and Browning note that even chaos when viewed from a particular level of abstraction, or a short enough time scale, will show predictability and deterministic properties. Chaos allows systems to evolve (or improve themselves) while order provides a unified, predictable and structured direction. "Complexity often concerns non-linear relationships between constantly changing entities. Systems theory, in contrast, studies static entities linked by linear relationships …complexity research concerns how complex behavior evolves or emerges from relatively simple local interactions between system components over time." (Manson, 2001)

To further explain IT complexity we borrow from Xia et al. and discuss various forms of structural and dynamic complexity. Structural complexity encapsulates relationships between organizational support elements (the human and management factors) as well as software specific IT elements: "diversity of user units, software environments, nature of data processing, variety of technology platform, need for integration, and the diversity of external vendors and contractors." Dynamic complexity includes "the rate and pattern of changes" in the organizational environment: "changes in user information needs, business processes, and organizational structures." It also includes the element's environment of "IT infrastructure, architecture and software development tools." (Xia & Lee, 2005)

By identifying challenges that SE must accomplish for complex information systems we get some insight on how to identify complex behavior at the enterprise level. Spaulding et al identified four SE challenges for complex information systems:

(1) Scale: The systems are large, in the number of components and sites and in the volume, variety, and velocity of information.
(2) Interconnections: The behavior of systems now emerges from the interactions among many components, generating behavior that is difficult to characterize.
(3) Changing Demands: The environment in which the systems operate requires adaptation on a small timescale compared with the time to develop and deploy systems by using the waterfall methodology.
(4) Evolving Technologies: Both hardware and protocols change rapidly in the information systems arena." (Spaulding, Gibson, Schreurs, Linsenbardt, & DeSimone, 2011)

A final concept of complexity theory is that of complex adaptive systems (CAS). This phenomenon explains the adaptability of a system to improve itself in relation to the environment and its structure for the greater good of the structure. "A CAS is an interconnected network of multiple entities (or agents) that exhibit adaptive action in response to changes in both the environment and the system of entities itself. Collective system performance or behavior emerges as a nonlinear and dynamic function of the large number of activities made in parallel by interacting entities. For example, the decisions made by firms facing imperfect local information, and variable environmental influences (demand) lead to a macro-level phenomenon (the bullwhip effect) (H. Lee, Padmanabhan, & Whang, 1997). Four mechanisms exist that produce a CAS: positive feedback, negative feedback, balance of exploration and exploitation, and multiple interactions. (Waldrop, 1992) "When all four of these mechanisms exist, then the self-organization of a complex system moves into the realm of a complex adaptive system. Without a controlling transition plan, the

dynamic characteristics of the enterprise information systems will develop through these CAS mechanisms (Honour & Browning, 2007).

Systems Engineering Tools

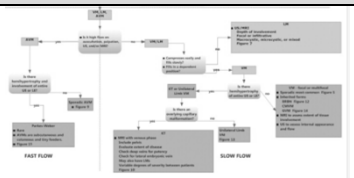
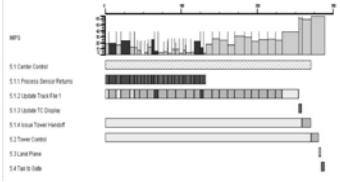
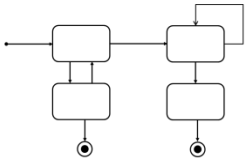
Many methods currently employed to harness complexity are explained in Systems Engineering Tools and Methods (Kamrani & Azimi, 2011): functional decomposition (hierarchy diagrams), object oriented descriptions (System Modeling Language (SySML), Unified Modeling Language (UML)), flow charts, functional flow diagrams, functional interface analysis (N2 diagrams), timeline diagrams, interface definition (input, output, signals, etc.), and dependency matrices (Pineda & Smith, 2011). With a goal to provide an easily understood depiction of systems in their context within an enterprise, graphically based SE tools are desirable. The tools mentioned in Table 3 can be grouped into two categories based on purpose. The first behavioral category is concerned with the flow of resources through a system or systems, and also depicting the dynamics, states and logic of a system. The second category is concerned with displaying the structure or hierarchy of a system and its system components. There is also structure in a system's data, and functionality.

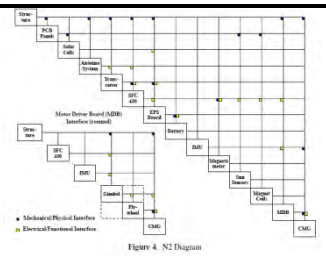
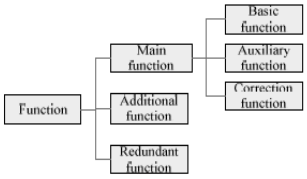
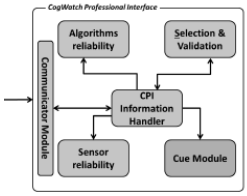
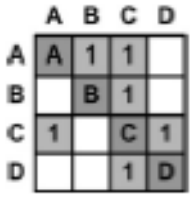
As the desired algorithm will be searching through an entire enterprise of IT systems and looking for redundancies across the global scale, the structural methods will be of most use as the behavioral methods do not capture this type of relationship between systems. Out of the structural methods in Table 3, only two methods, Design Structure Matrices (DSMs) and N2 diagrams, meet the needs of displaying the strength and frequency across systems. N2 diagrams have traditionally been used for static depictions, while DSM provides for

clustering, reorder, and iteration of change. DSM also allow for capturing and analysis of multiple relation characteristics (information flows, physical/logical connections, spatial). It is because of the dynamic attributes and the succinct graphical results provided by DSM that the authors feel that this tool represents the best approach for pursuing algorithmic development.

Due to the complexity of enterprise information systems, dependency matrices such as DSM should be a suitable technique. DSMweb.org defines DSM as "a simple tool to perform both the analysis and the management of complex systems. It enables the user to model, visualize, and analyze the dependencies among the entities of any system and derive suggestions for the improvement or synthesis of a system." In addition, DSMs allow mathematical manipulation of the relationships, which is conducive to the construction an automated roadmap algorithm.

Table 3. Systems Engineering Graphical Methods

Type of Method	**Systems Engineering Method**	**Example**	**Purpose (Pros/Cons)**
Behavioral	Flow Charts, Functional Flow Diagrams, Activity Models (UML, SySML), Business Process Modeling	 [Yu et al., 2012]	Provides a graphical depiction of the flow of a function or process. Shows the sequencing, the logic and the input/output flows. Does not articulate how systems or components are related physically or hierarchically.
	Timeline Diagrams, Timing Diagrams (UML), Sequence Diagrams (UML,SySML)	 [Jankovic, et al., 2012]	Depicts high level flow of process through time. May show messaging between components. Does not contain the detail of subsystems or components or their relations.
	Statechart, State Transition Diagram		Depicts the state of an object or activity, the events which trigger the transition and the rules/guards governing the transition. Also can capture activities/actions.

Structural	N2 Diagrams, Functional Interface Analysis,	 [Kasperek et al., 2014]	Displays the inputs and outputs between elements (functions, systems, components, etc.). Directional in nature. Does not show strength or frequency of relationship.
	Functional Decomposition	 [Tekes et al., 2014]	Classical approach to break complex systems into individual components. Does not show relationship between components/functions.
	Interface Definitions, Block Diagrams, Block Definition Diagrams (SysML) Object/Class Diagrams (UML)	 [Asundi and Fitz-Coy, 2013]	Describes how objects/components/systems connect with each other and allows for modularity. System/component/object centered. Lacks frequency, strength of interaction between all components/systems.
	Dependency Matrices (DSMs, interaction matrices, etc.)	 [Yu et al., 2012]	Displays relationships between elements (functions, components, systems, etc.). Can show strength/frequency of interaction. Requires substantial time/resources to construct as it can display large amounts of information in a single compact graphic. May require more than one matrix to describe all types of relations. May require previous decomposition as it often relates base level systems/components.

Literature Review and Results

Methodology

The following literature review methodology was employed:

(1) Literature in field reviewed for keyword identification
(2) Keywords verified through database queries
(3) Quality journals queried with final search string
(4) Abstracts reviewed to remove non-applicable articles
(5) Remaining articles identified for full text review

(6) Articles sorted by major concepts
(7) Articles studied and reviewed by sub-category

As systems engineering methods have been chosen to reduce complexity while focusing on the enterprise level instead of the typical product level it was imperative to explore the state of literature on DSMs and associated techniques. As DSMs have been employed under the guise of other names and techniques closely related to the DSM techniques described earlier, the following database search was conducted to determine the prevalent terms used and how much similarity exists between techniques. Names chosen for this search come from the DSM introduction section presented in journal articles (Browning, 2001b; Jankovic, Holley, & Yannou, 2012; Kasperek, Fink, Maisenbacher, Bauer, & Maurer, 2014) and through internet inquiry of related topics (DSMweb.org, Google Scholar, etc.). Please see Table 4 for results.

Table 4. Popularity of Terms used for Design Structure Matrix

Name Used	Number of Articles found with name in Title field	Number of Articles found with name in Keyword field
Dependency map	2	104
Dependency source matrix	0	1
Dependency structure matrix	11	156
Dependency structure method	0	0
Design precedence matrix	0	1
Design structure matrix	66	810
N2 diagram	0	46
Problem solving matrix (PSM)	0	38

It was found that many of the techniques were matrix-based, while others have predominately been used in other fields of study and bear little commonality with DSM methods. Names used for interaction study such as incidence matrix, interaction matrix, interaction maps appeared heavily in mathematical and natural science disciplines while the results for N2 matrices were firmly buried in chemistry as N2 represents nitrogen in the

periodic table. Of the three articles found with N2 matrix in the title all of them were referring to the study of nitrogen and the interaction between nitrogen and other elements.

As “design structure matrix” has become the dominant name for the type of problem solving required in the creation of a systems-of-systems level algorithm the literature was further queried for “design structure matrix” and its use at the system and product levels to further scope the literature.

Not surprisingly the product focus has been the dominant problem under study and only 119 out of 810 articles (roughly 15%) were found where “product” was not listed as a keyword. Not only does the inclusion of “product” in the search criteria result in a pool of data beyond the resources of the research team, but the problem under study is a system of systems where the relationships between higher level systems are of interest and not necessarily the relationships between the lowest level individual systems. The field of product component relationships has been well studied through the birth and growth of DSM methodology. This research is focused on a higher level of abstraction that has had little or no attention through the use of DSM methodology and restricting related research to what has been accomplished outside of the product focused arena is of primary interest. Please see Table 5 for query building word choice selection and results.

Table 5. Query Building Search Results

Search Criteria	Number of Works
"Design structure matrix" in Title	58
"Design structure matrices" in Title	6
"Design structure matrix" in Title AND "product" in Title	15
"Design structure matrices" in Title AND "product" in Title	1
"Design structure matrix" in Keyword	810
"Design structure matrices" in Keyword	142
"Design structure matrix" in Keyword AND "product" in Keyword	697
"Design structure matrix" in Keyword AND NOT "product" in Keyword	119
"Design structure matrix" in Title AND "product" in Keyword	38

Once the keyword search of choice was established as "design structure matrix" AND NOT "product" in author supplied keywords and again in the article titles, the following quality databases covering DSMs were queried directly from their websites: EBSCOhost's Academic Search Complete and Business Source Complete, IEEE Xplore, Thomson Reuters Web of Science, Science Direct, Wiley, and Springer (See Table 6). This initial broad search across multiple databases resulted in 82 duplicates indicating that the field of study should be well represented and not unduly influenced by database selection. Additionally a similar search was performed on Google Scholar which resulted in a similar list of articles as made through the combined list of articles in these databases.

Based on abstract reviews, articles were excluded where at least one of these conditions exists:

- Scope of research was narrowly focused (single product, process, project, etc.) instead of having an enterprise-wide application
- Coordinating project team members (coordination requires time based methods)
- Time or series-based (scheduling/sequencing)
- Risk management decision making (time based methods are used with risk)
- Does not mention DSM or other matrix methods in abstract (implies DSM is not a major focus of the research)

A second pass was conducted reviewing full text using the same conditions as the first pass and only a few more articles were excluded resulting in 47 articles for analysis. The

articles where then categorized with the following coding structure in regards to the research problem:

(1) Specifically mentions "system architecture" change, evaluation, or hierarchy abstraction/development
(2) Global or enterprise focus
(3) DSM + "other" methodology or theory
(4) Extensions to DSM methods/models/concepts
(5) Matrix-based algorithms
(6) Information Management (or software)
(7) Process improvement methods such as ERP/BPR
(8) Provides (or is in itself) a case study application of methods (does not include theoretical problems or examples unless example is from a real problem set)

The categories identified describe the body of literature reviewed and can further be associated with a higher level categorization of methods/concepts, theory, and applications. The three types of research approaches cross and complement each other and all three should be utilized to explore the creation of a roadmap generating algorithm (Fig 2). The higher order classifications allows for some generalization and understanding of how the body of knowledge connects what is known to the research goal of creating an impartial roadmap with respect to the chosen DSM systems engineering method (see Table 6).

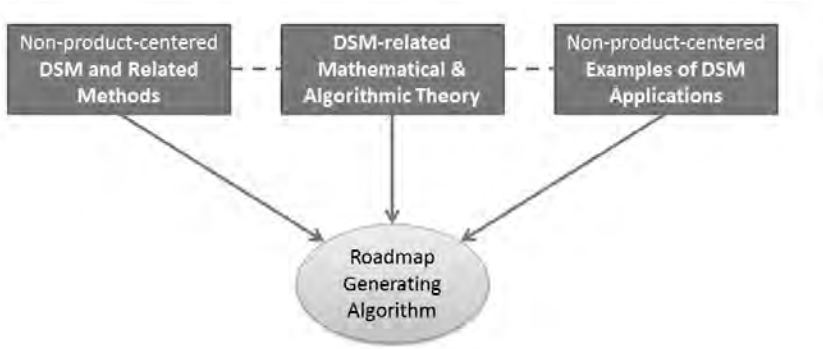


Figure 2. Relationship Diagram

The literature reviewed fell into three general categories which are interrelated and each will be relied upon to pursue the research goal of creating an automated algorithm designed to identify and cluster enterprise redundant systems.

Table 6. Distribution of Articles

High Level Category		Number and List of Articles
DSM Concept Development	DSM method extension	**19:** (Bartolomei, Hastings, de Neufville, & Rhodes, 2012; Brown, Nord, Ozkaya, & Pais, 2011; Browning, 2001b; Chang, Chiang, Wu, & Chang, 2011; Y. Chen, Cheng, & Yin, 2010; Dianting, Liu Yao-ming, & Lei, 2013; Guenov & Barker, 2005; B. Hamraz, Caldwell, & Clarkson, 2013a; Helmer, Yassine, & Meier, 2010; Holley, Jankovic, & Yannou, 2014; Kasperek et al., 2014; Lagerstrom, Baldwin, MacCormack, & Aier, 2013; Lambe & Martins, 2012; Lancaster & Cheng, 2008; Liang, 2009; Mikaelian, Rhodes, Nightingale, & Hastings, 2012; Nakata, 2009; Sharman & Yassine, 2007; van Beek, Erden, & Tomiyama, 2010)
	DSM + other methods/theories	**22:** (Aoyama & Tanabe, 2011; Bartolomei et al., 2012; Feng, Zhu, Sun, & Liu, 2011; Guenov & Barker, 2005; Holley et al., 2014; Koch, Maisenbacher, Maurer, Reinhart, & Zäh, 2014; Lambe & Martins, 2012; Li & Chen, 2014; Liang, 2009; Lopes, Bajracharya, Rashid, & Aksit, 2006; Luna, Lopes, Tao, Zapata, & Pineda, 2013; Malmstrom, Pikosz, & Malmqvist, 1999; McNerney, Farmer, Redner, & Trancik, 2011; Mikaelian et al., 2012; Sharman & Yassine, 2007; J. J. Simpson & Simpson, 2009; J. Simpson & Simpson, 2011; Tang, Zhang, & Dai, 2009; Tang, Zhu, Dai, & Zhang, 2009; van Beek et al., 2010; Yee, 2007; Yong-hui Guo, 2010)
Mathematic & Algorithmic Theory	Matrix-based algorithms	**10:** (Carriere, Kazman, & Ozkaya, 2010; Y. Chen et al., 2010; B. Hamraz et al., 2013a; Lagerstrom et al., 2013; Lambe & Martins, 2012; Li & Chen, 2014; Liang, 2009; J. J. Simpson & Simpson, 2009; J. Simpson & Simpson, 2011; Xiaogang, Chao, Jian, & Yahua, 2006)
	System architectural change / performance	**7:** (Aoyama & Tanabe, 2011; Brown et al., 2011; Carriere et al., 2010; Guenov & Barker, 2005; Lagerstrom et al., 2013; Luna et al., 2013; Sharman & Yassine, 2007)
Application	Global/enterprise focus	**10:** (Aoyama & Tanabe, 2011; Bartolomei et al., 2012; Bilalis & Maravelakis, 2006; Y. Chen et al., 2010; Feng et al., 2011; Jin-long Cao, Yi-yong Xiao, & Xiao-yan Xing, 2011; Lagerstrom et al., 2013; Luna et al., 2013; Nakata, 2009; J. J. Simpson & Simpson, 2009)
	Information Management or Software Development	**7:** (Aoyama & Tanabe, 2011; Brown et al., 2011; Kuqi, Eveleigh, Holzer, & Sarkani, 2012; Malmstrom et al., 1999; Wehrwein, 2013; Wen-Tin Lee, Kuo-Hsun Hsu, & Lee, 2012; Yong-hui Guo, 2010)
	BPR Application	**1:** (Feng et al., 2011)
	ERP Application	**None**
Provide case study data in examples		**26:** (Bilalis & Maravelakis, 2006; Brown et al., 2011; Browning, 2001b; Engel & Browning, 2008; Farid & McFarlane, 2006b; Hameri, Nihtila, & Rehn, 1999; B. Hamraz et al., 2013a; B. Hamraz, Caldwell, & Clarkson, 2013b; Helmer et al., 2010; Holley et al., 2014; Jin-long Cao et al., 2011; Li & Chen, 2014; Liang, 2009; Lopes et al., 2006; Luna et al., 2013; Malmstrom et al., 1999; Nakata, 2009; Sharman & Yassine, 2007; Tang, Zhu et al., 2009; van Beek et al., 2010; Wehrwein, 2013; Wen-Tin Lee, Whan-Yo Deng, Lee, & Shin-Jie Lee, 2010; Wen-Tin Lee et al., 2012)

Mathematical and Algorithmic Theory

Out of forty-seven articles only seven covered system architectural change and explored the depth of interconnectedness between elements (using various forms of DSM) and how variations in hypothetical changes in architecture can affect the performance of the system under study. Sharman and Yassine combined DSM clustering with Net Options Value (NOV) to provide a value measure of architecture in various states (Sharman & Yassine, 2007) while Lagerstrom et al. used the amount of interconnectedness to determine future propagation costs in subsequent software releases (Lagerstrom, Baldwin, MacCormack, & Aier, 2014). Highly interconnected programs required more time to integrate future changes, but the time to fully disconnect all dependencies prior to initial release was cost prohibitive. This resulted in the need to balance the right amount of interconnectedness in each software release. Evaluating changing enterprise architecture allows a CIO to determine if the change is beneficial to the enterprise at large. The iterative cycle that results from the software release study suggests that incremental and cyclic change is both measurable and beneficial. If this proves true for complex software, this methodology could benefit enterprise IS development.

Ten articles developed algorithms using DSM-based techniques or applied matrix-based algorithms before or after a DSM was constructed to perform their research. Li and Chen developed a clustering algorithm using matrix operations that can be applied to DSMs (and DMMs) in subsequent analysis (Li & Chen, 2014). Two articles by Simpson and Simpson present methods that can be combined with other systems engineering tools like DSM and N2 diagrams to address various types of complexity illustrating how DSM

methods can be applied to complex problems (J. J. Simpson & Simpson, 2009; J. Simpson & Simpson, 2011). Carriere et al. provide another architectural change measurement, however this one is built on a DSM dependent framework that was further examined using a developed cost metric (Carriere et al., 2010). Xiaogang et al. clustered a DSM and then applied their function to improve the DSM's results in regards to module identification (Xiaogang et al., 2006). Change propagation was examined by Hamraz et al. who used a matrix-based algorithm that employed matrices, DSMs, and matrix algebra to conduct their analysis (B. Hamraz et al., 2013a) and Chen et al. who constructed several matrix measuring relationships before combining those results to construct a DSM to measure change propagation (Y. Chen et al., 2010). These articles demonstrate DSM implementation at any stage of research (before, during, or after other major stages) and that DSM-based methods can be beneficial even if they are not technically a DSM. The process of building a DSM and viewing the relationships can be of use even before any sorting or clustering that is associated with a textbook DSM. Additionally, this research has shown that complexity can be addressed through DSM study and that the importance of module identification and change propagation should be taken into consideration when developing an enterprise clustering algorithm. Understanding why an algorithm clusters systems into modules and how other systems will be affected through the restructuring will be of utmost importance in future studies as there is little in case study literature examining the effects of complex organizational change from systems reduction.

DSM Concept Development

Of the 47 articles reviewed, 19 of them expanded on DSM methods by using DSM in a new fashion or developed internal construction methods while 21 articles combined DSM with other methods and theories that could provide insightful alternatives to DSM techniques already present in the literature. One article stands apart as it provides a launching point for this discussion and research; Browning's seminal 2001 article summarized and classified DSM methods and provided a reference article for the current state of DSM research prior to its publication (Browning, 2001b).

Many researchers in this review developed their own methods and algorithms finding new ways to employ DSM methodology. Liu et al. created the Linguistic DSM which uses qualitative variables to describe the strength of task relationships which can further be used in clustering highly related tasks, (Dianting et al., 2013) and van Beek et al. created a model that autonomously creates a weighted and clustered DSM based on initial conditions provided by the system designer (van Beek et al., 2010).

Other researchers took concepts developed by others and extended them further into the realm of DSM. Helmer et al. extended the work of (Pimmler & Eppinger, 1994) and (Sosa, Eppinger, & Rowles, 2003) by expanding on DSM entries to account for the requirements of interaction and spatial adjacency in a DSM designed to capture structural, energy, signal, material, and spatial interactions in each element.

Often research is about using tools and methods developed by others and finding a new use or new field of study to employ them. Kasperek et al. used complexity metrics such

as Karatkevich's Weight to analyze the effects of relational or structural design changes in DSMs of complex systems in early development (Kasperek et al., 2014), and Lagerstrom et al. used DSM and visibility of interactions to create an architectural visualization to reveal a "hidden structure" between software applications (Lagerstrom et al., 2013).

Some of the research combined the methods above and tied them into systems engineering theory and/or other theories resulting in a newly defined theoretical basis for the use of DSM. Aoyama and Tanabe combined DSM techniques with behavior in cyber-physical systems (Aoyama & Tanabe, 2011), Koch et al. drew on systems thinking to propose that the continuous manufacturing process is a system itself within an SoS which was then modeled with Multiple-Domain Matrices (MDM) for study in reconfiguration planning (Koch et al., 2014), and three studies used Real Options Theory (ROT) in tandem with DSM. Sharman and Yassine used a NOV algorithm from ROT to cluster DSMs as using ROT "proposes that design and industry evolution is an example of a *complex adaptive system*" (Sharman & Yassine, 2007), Mikaelian et al. used ROT to manage uncertainty in the use of DSMs to identify real options (Mikaelian et al., 2012), and finally, Engel and Browning used ROT to investigate adaptability in architecture options using DSM (Engel & Browning, 2008).

These extensions to DSM concepts show promise in building a roadmap generating algorithm as they:

- demonstrate the abilities to pull from qualitative variables
- autonomously create a DSM
- incorporate multiple requirements
- incorporate complexity metrics
- provide a visual aid to reveal hidden structure

- incorporate other theories in the development of algorithms, especially when dealing with complex problems such as those proposed in a SoS

Application Level Research

At the applications level of study it was noted that 26 of the 47 articles used case study data to convey and support their research. Only 10 of the 47 were externally focused on the effects to an entire company or across companies while none of them used the term "enterprise" to scope their efforts (Aoyama & Tanabe, 2011; Bartolomei et al., 2012; Bilalis & Maravelakis, 2006; Y. Chen et al., 2010; Feng et al., 2011; Jin-long Cao et al., 2011; Lagerstrom et al., 2013; Luna et al., 2013; Nakata, 2009; J. J. Simpson & Simpson, 2009). Only 7 articles (Aoyama & Tanabe, 2011; Brown et al., 2011; Kuqi et al., 2012; H. Lee et al., 1997; Malmstrom et al., 1999; Wehrwein, 2013; Yong-hui Guo, 2010) were used in the information management or software development sector and it was surprising to see only one article (Feng et al., 2011) mentioning BPR and not a single article mentioned ERP as the scope of this research is focused outside the product realm of DSM use.

From the articles with a global focus, Chen and Huang's article was of particular interest as it attempted to model supply chain dependencies using a DSM and was investigating System of Systems (SoS) outcomes in relation to system structure changes (Shi-Jie Chen & Huang, 2007). Although similar in focus as this paper's research, the theoretical model in (Shi-Jie Chen & Huang, 2007) was presented with a notional model and did not contain case study results as is common in the supply chain modeling field of study. Also of note is a paper presented at the 2014 Hawaii International Conference on Systems of Systems where the authors mentioned in their review of enterprise application architecture, "…we

have yet to see [DSMs] deployed in enterprise architecture modeling" and "[t]hese enterprise architecture approaches all rely on coupling and complexity measures to analyze architectures. None, however, uses DSMs to visualize the hidden structure of the architecture or to account for the indirect dependencies among software systems when measuring coupling." (Lagerstrom et al., 2013) Bilalis and Maravelakis were interested in the outcome of company performance (Bilalis & Maravelakis, 2006), Luna et al. used SoS performance to determine if new changes in behavior were acceptable in their methodology (Luna et al., 2013), Simpson and Simpson explain the payoffs of organizations and enterprises who build an SoS out of other systems at the expense of added complexity (J. J. Simpson & Simpson, 2009), Bartolomei et al. reiterates the need to take the "enterprise perspective" and uses that scope to employ multiple MDM and DSM matrices in their analysis (Bartolomei et al., 2012), Cao et al. use coupling costs to identify the influence on global performance in estimating life cycle cost (Jin-long Cao et al., 2011), and finally Aoyama and Tanabe viewed the SoS of automobile systems and their interrelations and effects on the global behavior of the vehicle (Aoyama & Tanabe, 2011). The information management and software related articles consisted of software design/redesign using DSMs or related matrices to visualize the relationships (Aoyama & Tanabe, 2011; Brown et al., 2011; Wehrwein, 2013; Wen-Tin Lee et al., 2012), improve software usage based on human interaction with software displayed via DSM (Kuqi et al., 2012), and creating models based on information flow using DSM and IDEF0 (Malmstrom et al., 1999; Yong-hui Guo, 2010).

These application level studies illustrate the need and the usefulness of DSM in various non-product-centered studies, however only a small number were focused on the

global aspect of change resulting from DSM restructuring. The small sample indicates that this aspect of DSM study has only begun. Specifically, enterprise level SoS study in DSM and their relationship to ERP application is virtually non-existent and is rich for future research. A gap exists in the research where all of these concepts have yet to be applied in a single study on an enterprise or other macro-level complex SoS.

Trends in Literature

There was minimal indication of potential trends in the literature other than the expected growth from simple application of DSM to the more advanced intersection of methods and theories from other disciplines. The one exception is the apparent upward trend in applying DSM to various aspects of change: change propagation, change management, and structural design change were frequent topics from 2012 to 2014. As this was a focused literature review the authors hesitate to comment on possible trends in the more general field of DSM research. Please refer to the Appendix (available by request) for the complete chronologically sorted list of articles reviewed in this study.

Conclusion

Discussion

A systematic literature review was conducted to frame a managerial problem in terms of systems science and engineering, the systems engineering methods and tools that could be used to create an automated roadmap generation algorithm, and to what extent past DSM research has been applied in a macro-level context. This paper described a complex problem that is arguably the result of the evolutionary process of business management and presented

complexity and systems theories to conceptualize and understand the current state of the problem. This paper then presented a list and review of methods that are used in systems theory to harness complexity and study the problem with appropriate methods for the goal of designing an algorithm that can assist managers of complex systems to understand the complexity of their systems-of-systems.

The methodology began with conducting a search using key terms used in the DSM field to focus the research, but also to provide a review of the prevalence of key terms in the literature for the benefit of fellow researchers. Once the search methodology was completed, a conceptual framework for the literature found was constructed. Extant research was eventually grouped into three major categories: theory development, methodology development, and applications of theory and methodology.

Some of the most important discoveries from the literature review assist in scoping and defining the problem, showcasing methodologies applied to similar problems, and providing a basis for future research. As mentioned earlier, the seminal 2001 article by Browning provides history, support, and the basis for virtually any new DSM research as it reviewed and classified each of the general DSM types that have subsequently been studied and expanded by others (Browning, 2001b). The research presented by Helmer et al. expands on the basics described by Browning and provides a comprehensive review of clustering methods and provides a DSM method using five-dimensions in each element and a means to cluster the DSM for identifying modules and interfaces which could be instrumental in the development of a roadmap generating algorithm (Helmer et al., 2010). Expanding the focus of DSM, Chen and Huang studied supply chain structure using DSM methods using a

notional problem set which could potentially assist supply chain managers in decision making in regards to dealing with other companies (suppliers or customers)(Shi-Jie Chen & Huang, 2007). Researching supply chains in a DSM context is one of the few examples of DSMs employed in a high-level context that reaches across system boundaries which is the focus of this and subsequent research. Finally, also mentioned earlier, Lagerstrom et al. stated in their review that "we have yet to see [DSMs] deployed in enterprise architecture modeling" before they searched for the hidden structure of software architecture in an enterprise setting, which is very closely related to the research presented in this paper (Lagerstrom et al., 2013). While (Lagerstrom et al., 2013) was searching for the hidden structures, this research is looking for a method to identify macro-level service redundancy across major enterprises (100s to 1000s of systems) for the purpose of consolidation and redundancy removal (and thus a reduction in complexity).

The conclusion is that DSMs should provide a suitable technique for the creation of a roadmap generating algorithm for the purpose of reducing enterprise information systems complexity. To investigate this proposition a basic algorithm is proposed in Figure 3.

Loop until convergence (enterprise reduced to one ERP).
Step 1. Map the relationship between Enterprise systems in a matrix
Step 2. Cluster systems
Using measure of similarity and/or relationship strength, implement a cost function that penalizes large clusters while rewarding cluster membership.
Step 3. Reorder the DSM placing clustered systems together
Step 4. Integrate clusters into single systems while retaining relationships with other clusters
Step 5. Create a new, smaller DSM based on the integrated clusters
Step 6. Record each iteration providing the roadmap to evolve a complex legacy enterprise

Figure 3. Basic roadmap producing algorithm

Limitations and Future Research

This literature review was restricted to the following databases and specific search criteria used. The initial set of databases queried for the purpose of settling on specific search terms was: Academic Search Complete, Business Source Complete, IEEE/IET, MasterFILE Premier, and Web of Science. The second set of databases used for the literature review was: EBSCOhost, IEEE Xplore, Thomson Reuters Web of Science, Science Direct, Wiley, and Springer. Selection criteria were first accomplished through database query and Meta data review followed by a selection and analysis by a single author while further review and analysis was conducted by each of the authors.

Future research will identify the algorithm to automate the creation of an architectural transition roadmap, from the "As-Is" and to targeted reductions over time. The basic algorithm will be explored in further detail and programmed for future modeling and simulation. After researching the growth of enterprise redundancy, System-of- Systems, and

systems engineering methods, the authors propose DSMs should provide a means for developing a roadmap generating algorithm.

IV. Collapsing Design Structure Matrices for Enterprise Integration Planning

The Abstract – Large enterprises have grown in complexity over the last few decades and the integration of redundant systems has become a major challenge for organizations desiring a leaner, less wasteful, systems architecture. Traditional design structure matrices (DSMs) have been studied extensively for analyzing and clustering complex interacting components within a product or system. To address enterprise-level issues, this paper introduces a method to use traditional DSMs to identify and integrate redundant enterprise information technology (IT) systems. The proposed algorithm performs an iterative reduction on a collection of systems resulting in an integration plan for enterprise stakeholders, a task which is highly relevant when evolving legacy IT towards fewer, or a single, integrated solution.

Keywords: System analysis and design; enterprise architecture; design structure matrices (DSM); information systems; systems integration

Introduction

The merging of large companies and the forming of large enterprises in today's marketplace is rather common. However, the true merger of corporations is fraught with complex information technology (IT) decisions, such as how to best merge incompatible databases for the benefits of shared information across business functions. There are potential solutions available to this problem, but often these solutions are constrained due to time, money, and experience. One method that has had mixed results (Ahmad & Cuenca, 2013; Motwani et al., 2005; Umble, 2003; Wang et al., 2008) is enterprise resource planning (ERP). ERPs replace incompatible legacy systems with a single system that requires strongly coupled data and business process logic (Baharum, Ngadiman, & Haron, 2009).

One of the big challenges of an ERP or other business process reengineering (BPR) effort is the implementation plan. In 2002, the Dept. of Homeland Security was charged with

evolving over 500 financial systems into one system (U.S. Government Accountability Office, 2007; U.S. Government Accountability Office, 2013). This attempt, and the next two ERP attempts failed. They are currently on their fourth attempt at an integrated financial, asset management and acquisition management system. Likewise, the US Air Force attempted one of the largest ERP implementation, the Expeditionary Combat Support System (ECSS), designed to incorporate over 250 into one. After ten years and over a billion dollars, the program was cancelled. There were many reasons ECSS failed (Reilly) and one of the reasons listed (Bliss, 2013) was the lack of a unified plan with frequent short term milestones upon which to access progress. In essence, ECSS lacked a detailed roadmap to transition the myriad of IT systems, their functionality and data, toward the "To-Be" architecture.

This paper introduces a methodology called Collapsing Design Structure Matrices (C-DSMs) to automatically produce such a roadmap, which embraces the iterative development approach in the software industry. The methodology makes use of an algorithm that generates an integration roadmap based on user preferences and constraints. Prior to exploiting this algorithm, a user maps the relationships between the enterprise IT systems in a matrix (or set of matrices) and then specifies the controlling reduction parameters for each iteration. The algorithm clusters and integrates systems based on the strength of relationships and other financial characteristics. Through user-controlled iterations, the number of systems optimally collapses resulting in fewer and fewer enterprise systems. This reduction is similar to legacy IT cutover strategies, where both persistent and transactional data is converted to a standard format, and functionality subsumed by a modern ERP.

The remainder of this paper is organized as follows. Related work in the field of DSMs is covered in Section 2. Section 3 presents the algorithm and identifies and defines the relationships and possible characteristics between DSM elements. In Section 4, the results of the examples and experiments are presented to validate the model. We conclude with Section 5 with utility and application discussions, algorithm limitations and opportunities for future research.

Literature Review

Enterprise Information Technology

As the enterprise IT infrastructure expands, it becomes more challenging to understand the totality of the relationships between systems. To make matters worse, enterprises that have grown over decades of technological innovation are riddled with legacy IT systems that have grown separately in disconnected silos, but must now must better integrate and share information. Fu et al. (Fu Xiang-ling et al., 2010) studied these problems and suggest some solutions to cope with these "islands of information". Starting with a new unified data system, one should build a unified "nervous system" to integrate the islands, or integrate groupings of systems that may cross departments (Fu Xiang-ling et al., 2010). Out of necessity, many organizations have attempted many self-improvement techniques such as Business Process Reengineering (BPR), Total Quality Management (TQM), Lean, Six-Sigma, or implementing an ERP with mixed results.

An enterprise resource planning (ERP) system is a cross-functional enterprise system driven by an integrated suite of software modules that supports the basic internal business

processes of a company. An ERP system gives an organization an integrated real-time view of its core business processes such as production, order processing, and inventory management, tied together by ERP application software and a common database maintained by a database management system. ERP systems track business resources (such as cash, raw materials, equipment, people, and production capacity) and the status of commitments made by the business (such as project execution, purchase orders, and human resource activities), no matter which department (finance, contracting, individual sub-organizations, and so on) has entered the data into the system. ERP systems facilitate information flow between all business functions inside the organization and manage connections to outside stakeholders" (Bidgoli, 2004).

While there have been some successful ERP implementations, there have been far more failures and many others demonstrated less than the expected return on investment (ROI) (Umble, 2003). A failed ERP implementation hurts the implementing organization in at least three ways: cost of development and implementation up to the point of failure, reinvestment costs in legacy systems to implement currently needed capabilities, and continued cost of unrealized efficiencies (Baxter, 2010).

Some of the most difficult enterprises to transform are government agencies. The RAND Corporation recently investigated the conditions for successful ERP implementation and the first condition for successful change was the business case. This factor was used to determine if the transformation endeavour should be initiated. It is contingent upon a clear current "As-Is" environment and the target "To-Be" architecture to achieve enterprise cost and performance goals (Riposo et al., 2013). Several studies have captured ERP

implementation lessons learned (Al-Fawaz, K., Al-Salti, Z., & Eldabi, T., 2008; Baxter, 2010; Chen, C. C., Law, C. C., & Yang, S. C., 2009). In particular, Al-Fawaz (Al-Fawaz, K., Al-Salti, Z., & Eldabi, T., 2008) conducted a comprehensive review of available literature, and identified the most cited ERP critical success factors; two of his factors included business planning and ERP selection.

Enterprise architects need a method or tool that will help generate a roadmap, allowing depiction of the current state (the "As-Is"), the desired state (the "To-Be"), and incremental transitions. There are standard representations for developing the As-Is and the To-Be architectures, supported by various architecture frameworks, such as the Zachman framework (Zachman, 2010), the Open Group Architecture Framework (TOGAF), the Department of Defense Architecture Framework (DoDAF), and several others. However, there is little in the literature explaining how transition planning should be accomplished. Currently, many of the techniques employ expert opinion and "gut" decisions by process owners leading to varying results. The literature review will support the proposition that DSM methods are suitable for developing roadmap-generating algorithms to reduce redundancy and complexity of enterprise information systems.

Design Structure Matrices

The design structure matrix (DSM) is a representation and analysis tool for systems modeling, especially for purposes of analysis and integration. DSMweb.org defines DSM as "a simple tool to perform both the analysis and the management of complex systems. It enables the user to model, visualize, and analyze the dependencies among the entities of any system and derive suggestions for the improvement or synthesis of a system." As a matrix,

the DSM captures the relationships between components of a system, or systems themselves, across the rows and columns (see Figure 4). In addition, DSMs allow mathematical manipulation of the relationships, which is conducive to the construction of an automated roadmap algorithm.

	A	B	C	D
A		1	1	
B			1	
C	1			1
D			1	

	A	B	C	D
A		4.7	5.3	
B			2.1	
C	-1			0.8
D			-2.2	

Figure 4. Example DSM capturing binary relationships (left) or strength of relations (right) between components or systems.

In previous work by the authors, it was found that the preponderance of literature on DSMs have been focused on product development (697 of 810 references). The DSM presented by Steward (Steward, 1981) to analyze complex systems in a matrix format provides the foundation for this paper. Although originally focused on products or physical components, DSMs have also been extended into organizational DSMs, multi-domain-matrices (MDMs), and process DSMs (Eppinger & Browning, 2012) that allow exploration of organizational, social or team relationships and structure (Batallas & Yassine, 2006),(Bartolomei et al., 2012). Another recent DSM effort proposed an approach to investigate risk assessment in the software design process (Fu, Li, & Chen, 2012). Traditional DSMs have not been used to propose the deletion or removal of systems or as a systems integration tool to provide an iterative enterprise integration plan.

The algorithm presented in this paper extends previous work by Thebeau (Thebeau, 2001) who clustered DSMs based on algorithms identified by Carlos Fernandez. This algorithm implements a multiple objective clustering, based on relationship strengths (both intra and inter clusters) to find the optimal solution. Thebeau's clustering approach is modified to allow for longitudinal rework of a collection of enterprise IT systems, described by either relationship similarity or cost to integrate. The enterprise IT landscape "collapses" over time, to reduce redundancy and save operations and maintenance costs.

DSM factors

Design structure matrices (DSM) have been used for a variety of applications, and thus, the interactions between elements (row and column) model varying characteristics. Such applications include physical components of a system architecture, organizational/team-based DSMs or activity/process DSMs (Browning, 2001b). As such, the interactions (values in the matrix cells) may capture spatial, energy, information or material relationships. In fact, multi-dimensional matrices could capture several types of interactions between systems (Helmer et al., 2010).

For enterprise information system evolution, such relationships could include the following taxonomy in Table 7.

Table 7. Taxonomy of Enterprise Information Systems DSM relationships

Type	Example Dependencies
Structural	• # of Interfaces/Interface control documents (ICDs) • Interface complexity/scope (P2P*, EAI, batch/ real-time, loose coupling, service encapsulation, standard implementation) • Projected Lifespan (how long is it required) • Maintainability, Adaptability, Flexibility, Amount of change
Functional	• Shared functionality
Informational	• Number of information exchanges (in/out) • Frequency of exchange (daily, real-time, monthly) • Diversity of exchanges (transaction types, batch) • Volume of data across the interface • Common data elements • Interoperability
Implementational	• Commonality of Programming language • Likelihood of successful integration • Performance requirements (Service level agreements)
Financial	• Cost to integrate/ Modify • Cost/ schedule to translate data

* Point-to-point interface (P2P), Enterprise Application Interface (EAI)

In addition to the relationships between systems, systems may also have individual characteristics that may need to be incorporated into the transition plan. Such factors could include Operations and Maintenance Cost (annual), priority/mission requirement, or criticality.

Collapsing DSM Methodology

The static DSM as presented by Browning (Browning, 2001b) serves as the representation where rows and columns represent enterprise information systems and the matrix entries represent the strength of the system relationships. Our model also makes use

of the diagonal to hold previous algorithmic calculations from previous iterations to ensure the collapsed DSM has retained all of the original DSM's relationships. Some of the preliminary definitions are provided in Table 8.

Table 8. Collapsing DSM definitions

System X_i: one of the $i=1..n$ Enterprise systems
x_{ij}: relationship, strength or relation, or cost of integration between systems i and j
Clusters k: Clusters represent a group of highly related systems that are "most alike" which indicates higher likelihood of compatibility for the purpose in integration. With greater compatibility there should be less difficulty and cost in the integration
Total Coordination Cost: Objective function that quantifies the cost of clustering systems, penalizing for too much inter-cluster relations and cluster size

The objective of this algorithm is to minimize the number of relationships that are not in a cluster through clustering related systems, integrating (or collapsing) those systems into a single system to produce a smaller, less complex DSM. For each iteration of the algorithm a clustered DSM and two "collapsed" DSMs are created. The process is repeated until the user has reached their desired level of reduction, or until the DSM has collapsed to a single system. Each iteration represents a time period specified by the user and the amount of reduction desired and number of systems to integrate per iteration can be tailored accordingly. If one iteration is a single year and the systems are complex, cluster size and overall enterprise reduction should be small. The smaller, more frequent, increments approach is the driving methodology behind our approach as it lends to more frequent progress checks with decision makers.

Assumptions

The algorithm assumes that the identification of a relationship, or multiple relationships, between systems is indicative of compatibility. In the binary case there is no

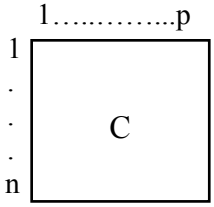
qualification as to the amount or strength of the current relationship. Non-binary methods could include normalized Likert scales, weighted sums of attributes, or simply a percentage of similarity between the two focus systems. The type and range of numbers used is a decision for system designers charged with populating an enterprise DSM. This research utilizes matrices with the understanding of some measure of similarity between two systems. Similarity can be the physical or logical connection between two systems or it can be some other measurement as to the amount two systems have in common. For one type of DSM proposed, the measurement of the relationship is expressed as the onetime cost to integrate the two systems of interest. Other DSMs can utilize a normalization technique to combine a series of weighted Likert scaled responses by system designers. Any reasonable metric can be used to fill in the matrix.

Cost within the algorithm is not the same as monetary costs that may be entered into the DSM. Cost for purposes of the clustering routine includes algorithmic penalties to manipulate whether elements are placed in the same cluster or not. Algorithmic cost represents how difficult one element or cluster is to integrate with another, once clusters begin to grow as large cluster sizes will be harder to integrate (cost in time and complexity) and too many small clusters will leave too much complexity in the overall DSM. Algorithmic cost will be attributed to constraints on cluster size via penalties in either extreme within the clustering algorithm. Monetary cost can be used to construct a negatively charged DSM, but monetarily based costs and calculations are not built into the algorithm.

Decision Variables

Matrix **C** = $\{c_{ik}\}, i = 1..n, k = 1..p$ determines which system *i* is assigned to each cluster *k*.

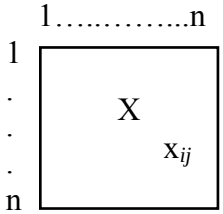
$$= \begin{cases} 1, & \text{if system } i \text{ is in the cluster } k \\ 0, & \text{otherwise} \end{cases}$$



Parameters

Given *n* number of systems, a constructed Design Structure Matrix **X** = $\{x_{ij}\}$, *i, j = 1..n*, where:

$$x_{ij} = \begin{cases} \neq 0, & \text{if system } i \text{ has a relationship with system } j \\ 0, & \text{otherwise} \end{cases}$$



Likewise, the relationship may reflect an amount or strength of relationship

$\min\_x \leq x_{ij} \leq \max\_x$,
if system *i* has a bounded quantifiable relationship with system *j*

Lastly, the relationship may reflect a set of relationships

$\min\_x_m \leq x_{ijm} \leq \max\_x_m$,
if system *i* has *m* bounded quantifiable relationships with system *j*

Algorithm

Thebeau produced the following Fernandez algorithm in MATLAB and is modified in this research to produce an integration plan of optimal reductions (Thebeau, 2001):

1. Each element is initially placed in its own cluster
2. Calculate the Total Coordination Cost of the Cluster Matrix
3. Randomly choose an element *i*
4. Calculate bid from all clusters for the selected element *i*

5. Randomly choose a number between 1 and *rand_bid* (algorithm parameter)
6. Calculate the Coordination Cost if the selected element becomes a member of the cluster with highest bid (use second highest bid if step 5 is equal to *rand_bid*)*
7. Randomly choose a number between 1 and r*and_accept* (algorithm parameter)
8. If new Coordination Cost is lower than the old coordination cost or the number chosen in step 7 is equal to *rand_accept*, make the change permanent otherwise make no changes
9. Go back to Step 3 until repeated a set number of times*

*Steps 6 and 8 use simulated annealing by randomly accepting the second highest bid (6) and accepting changes even if solution worsens (8). This helps avoid finding local optima.

In Step 3 an element *i* is chosen and the clusters make bids for the element where the following calculation takes place:

$$ClusterBid_{ki} = \frac{inout_{ki}{}^{powdep}}{ClusterSize_k{}^{powbid}}$$

Equation 1

Where:

k	= cluster number
$ClusterBid_{ki}$	= Bid from Cluster *k* for the chosen element *i*
$inout_{ki}$	= sum of DSM interactions of the chosen element *i* with each of the elements in cluster *k*
powdep	= exponential to emphasize interactions
powbid	= exponential to penalize size of the cluster
$ClusterSize_k$	= size of cluster *k*

Objective Function

The objective is to minimize the absolute value of Total Coordination Cost through re-clustering. The absolute value is needed depending on the factors used to describe the relationships between elements in the DSM. Some factors have a positive scale

(coordination cost/ similarity, integration costs), and others have a negative scale (operations, repair, and maintenance cost savings).

$$Total\ Coordination\ Cost = \sum_{k\ =\ 1..z} \sum_{ij\ =\ 1..n} \left(ExtraClusterCost_{ij_k} + IntraclusterCost_{ij_k}\right)$$

Equation 2

Where:

$$ExtraClusterCost_{ij_k} = \begin{cases} (x_{ij} + x_{ji}) * n^{powcc} & \text{if } c_{ik} \neq c_{jk} \\ 0, & \text{otherwise} \end{cases}$$

$$IntraClusterCost_{ij_k} = \begin{cases} (x_{ij} + x_{ji}) * ClusterSize_k{}^{powcc} & \text{if } c_{ik} = c_{jk} \\ 0, & \text{otherwise} \end{cases}$$

z = number of clusters in the solution
n = number of elements (number of systems in the matrix)

The size penalty (*powcc*) penalizes large clusters and allows the designer to further penalize or remove the penalty by setting this to 1.0 (default value). The original algorithm referred to these calculations as costs; we will clarify these as clustering costs.

Constraints

The original algorithm was an unconstrained multiple objective formulation. The extensions for iterative use in element reduction (collapsing) resulted in adding user constraints on max cluster size τ, minimum number of clusters δ, and maximum number of clusters p_2. These constraints prevent the algorithm from integrating beyond the capabilities or resources of the organization during each iteration. Typically, clustering routines in static DSMs allow elements to belong to more than one cluster. For example, if a driveshaft connects two different components the driveshaft would be the overlapping element in two clusters (Browning, 2001b). Given the goal of integration of large numbers of physical systems, multiple cluster memberships are not allowed. This departure from the original

algorithm is enforced through an added constraint in the clustering function and solutions with multiple cluster memberships are rejected. The constraints are simply:

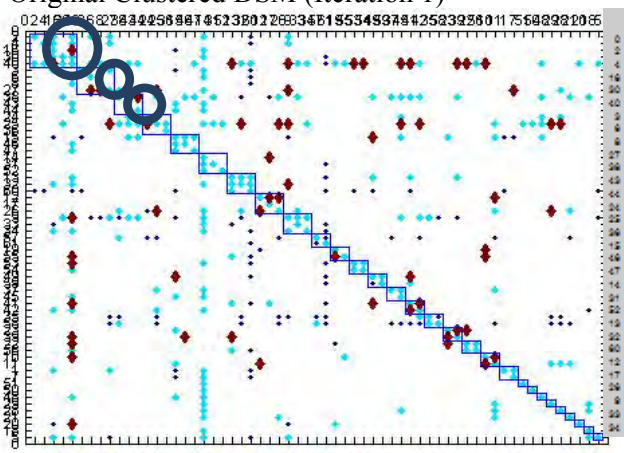
Table 9. Table of Constraints

$\sum_{l=1}^{m} c_{lk} \leq \tau$	$\sum_{k=1}^{p} c_{ik} \geq \delta$
$\sum_{k=1}^{p} c_{ik} \leq \varepsilon$	$\sum_{i=1}^{p} c_{ij} = 1$

Collapsing DSM Algorithm

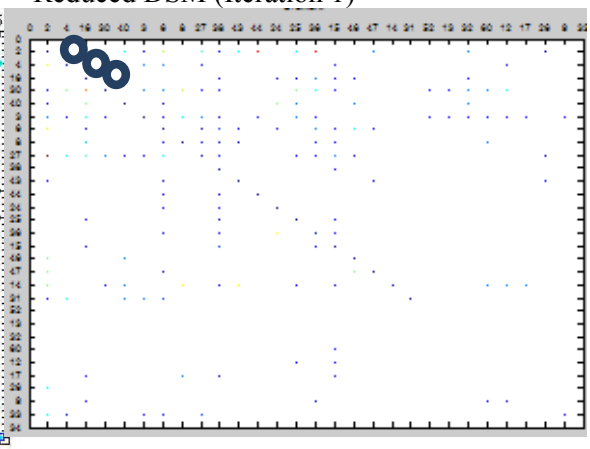
Once Thebeau's algorithm produced a clustered DSM that met the original objective function of the lowest metric solution while meeting the user-defined constraints we introduced, a second DSM is constructed to represent the next evolution of the DSM. To produce a smaller "reduced" DSM the clusters are integrated into a single system and the new DSM size is equal to the previous solution's number of clusters. To ensure relationships and relationship strengths (in non-binary DSMs) were not lost, the clustered DSM's metric is compared to the reduced DSM's metric. Even though the dimensions of the DSMs are different, the metrics are identical as the ExtraClusterCosts are recorded in the off-diagonal elements of the reduced DSM and the IntraClusterCosts are recorded on the diagonal of the reduced DSM. In Figure 5, the same three clusters of the original DSM and the associated new elements in the reduced algorithm are highlighted.

Original Clustered DSM (Iteration 1)



Number of original systems: 61
Total Coordination Cost: 1,091,006

Reduced DSM (Iteration 1)



Number of systems after collapsing: 29
Total Coordination Cost: 1,091,006

Figure 5. Original DSM Reordered by Clustering and First Reduced DSM (Iteration 1)

As an analogy, the DSM appears to be "folding in" on its diagonal during each iteration, without reducing the actual functionality or relationships represented by the original DSM. Relationships do not have to be binary and can represent a variety of relevant metrics. In this example, relationship strengths range from 0.0 to 2.0. The user defined inputs were max cluster size 10 and an overall system reduction of 50%. The actual percent reduction achieved was 52.5% in one iteration of the algorithm. Although graphically represented with sized diamonds, the DSMs have numeric entries. The clustered DSM holds the relationship values and the first Flattened DSM contains the combined costs of clustered systems C_{ik} and C_{jk}. Complimentary to the creation of the first penalty-retaining reduced DSM, a second reduced DSM is created which retains the combined relationship values. This second reduced DSM provides the starting DSM for the next iteration of the algorithm, which will lead to yet another set of smaller matrices based on the constraints provided by the user.

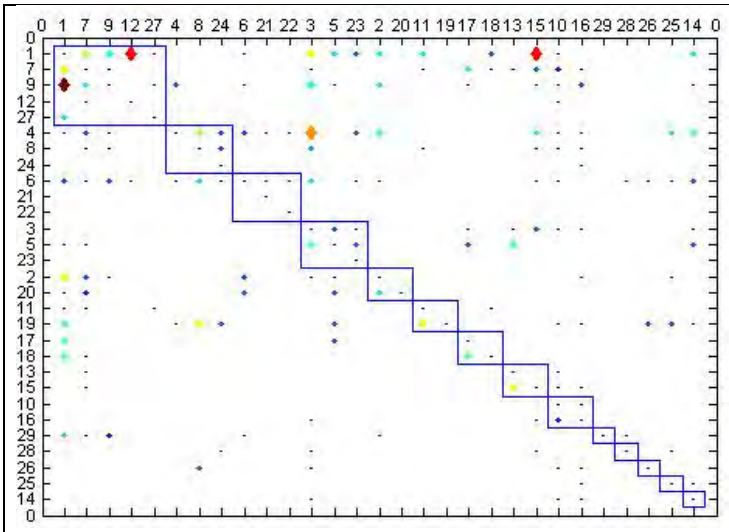
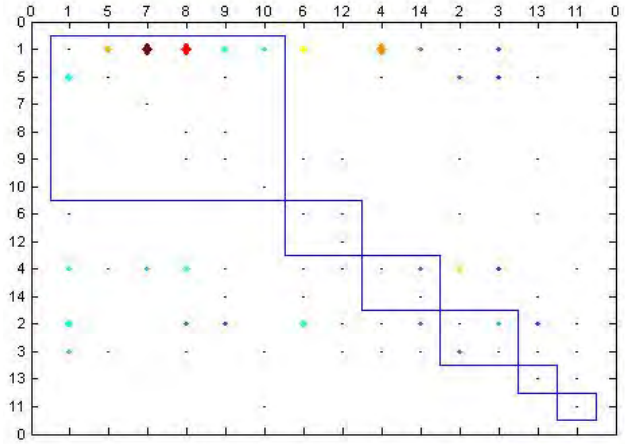
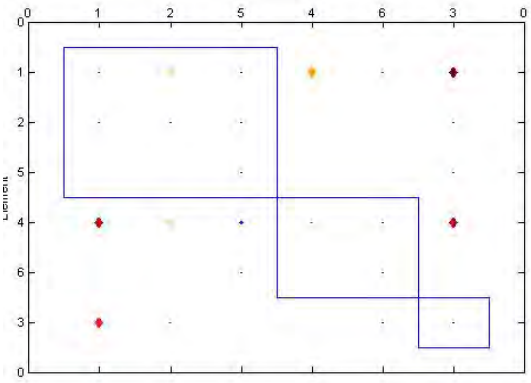
	Iteration: 2 Number of systems before collapsing: 29 Percent Reduction Desired: 50% Max Cluster Size: 10
	Iteration: 3 Number of systems before collapsing: 14 Percent Reduction Desired: 50% Max Cluster Size: 10
	Iteration: 4 Number of systems before collapsing: 6 Percent Reduction Desired: 50% Max Cluster Size: 5

Figure 6. Iterations 2-4 of the collapsing DSM algorithm

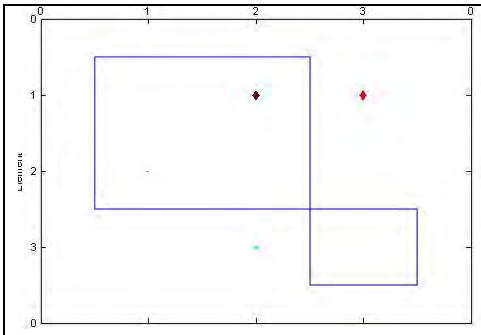
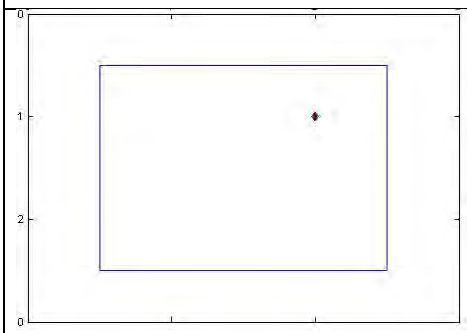
	Iteration: 5 Number of systems before collapsing: 3 Percent Reduction Desired: 50% Max Cluster Size: 5
	Iteration: 6 Number of systems before collapsing: 2 Percent Reduction Desired: 50% Max Cluster Size: 5

Figure 7. Iterations 5 and 6 of the collapsing DSM algorithm.

After six iterations of folding on the diagonal, the DSM was reduced from 61 systems to 1. Each iteration was controlled by the user employing a constraint of 50% overall DSM system reduction. The first three iterations employed a constraint of no more than ten elements per cluster, while the last three iterations employed a maximum of five elements.

Clustering Cost

The best solutions, with the lowest total coordination costs, are when elements with strong relationships are clustered together. Like many stochastic or heuristic search algorithms, one observes both 1) long sequences of epics with ”flat” objective values, that indicate better solutions were not found, and 2) “spikes” in the objective value, that

represent epics where simulated annealing calculated and selected solutions that were not optimal for the purpose of avoiding local minima (please see Figure 8).

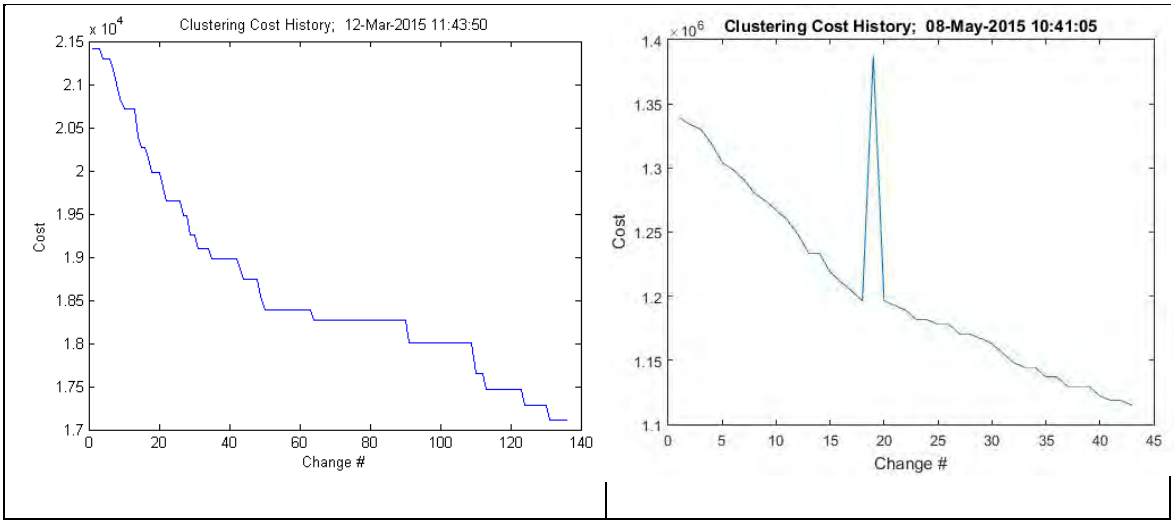


Figure 8. Examples of the Clustering Cost History using simulated annealing

Integration Cost

Integration cost represents an enterprise's monetary investment in combining two or more systems. Estimating the cost of a future combined system without any knowledge of what the future system's capabilities will be is modeled by simply taking the average of the currently known systems that are clustered for integration. In an application setting, users will provide actual estimates before each iteration, but these simple average estimates can be used for long-term planning. As with any forecasting technique it should be understood that there is less reliability the further the projection calculates from the current iteration.

To conclude and illustrate the algorithm for the purpose of generating a roadmap for decision-makers, see Figure 9.

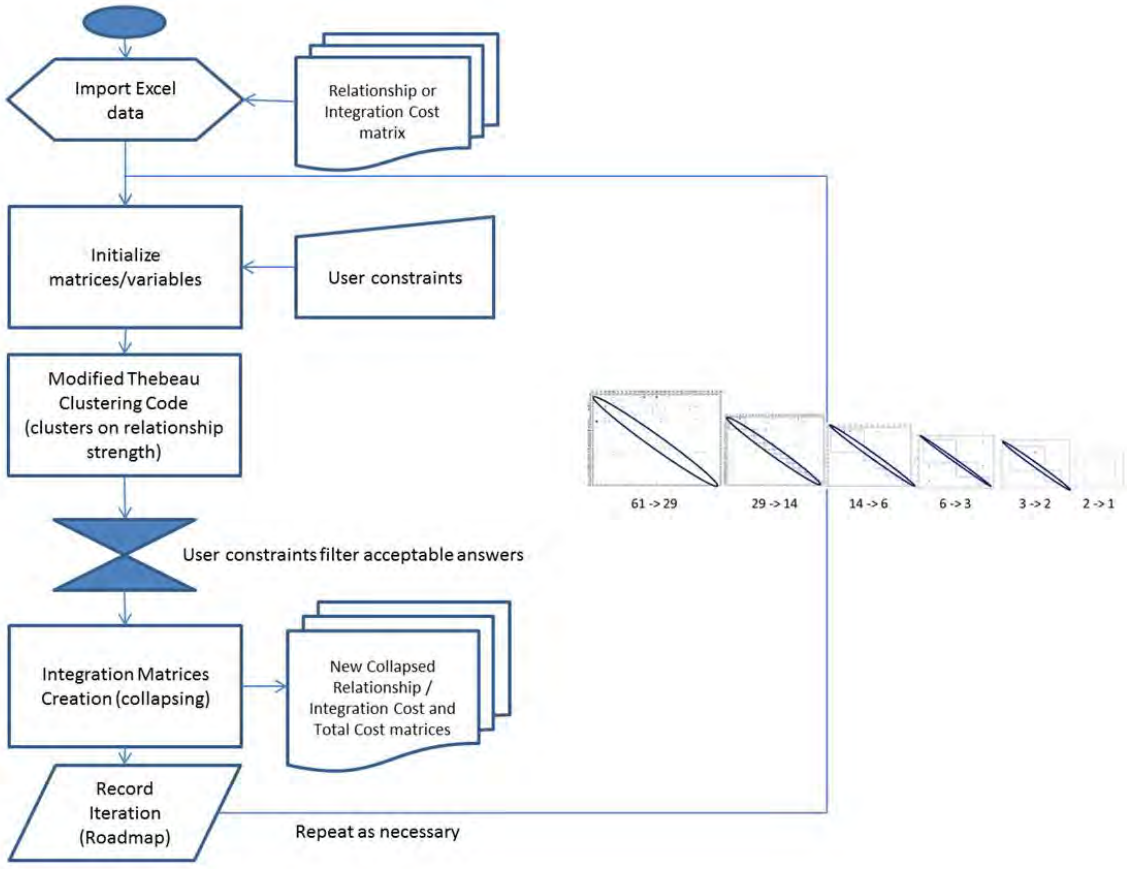


Figure 9. Collapsing DSM Algorithm Flowchart

Illustrative Example of an Enterprise Application

To examine the validity and utility of the algorithm, a sample Enterprise was generated, where each cell represents a system's estimation of integration cost. Integration cost was chosen as it is a single factor that can represent a large variety of other factors that an enterprise can use to describe the complexity between any two systems. Regardless of which factors are used to represent the similarities or differences between systems (such as database language, data types, age, functionality, etc.) eventually those factors are reduced to a monetary value of cost. As cost can be estimated for any pairing of systems a fully

connected matrix is created which can be problematic for some algorithms. To generate random numbers that would cover a large range of cost estimates without resulting in an overall uniform distribution, a fairly flat problem space, a chi-squared distribution with two degrees of freedom was chosen as it produced a more volatile collection of numbers, with the majority of the numbers amassed near the origin. The values initially ranged from $0.004M to $151M and the same matrix was used for all runs for the remainder of the experiment. In this case, $0.004M represents the least amount of money required to integrate (integration cost cannot equal zero), while $151M indicates that the systems are least similar and would require the most amount of resources to integrate. Cost is represented as negative values, though cell values can also be positive to represent the strength of similarity between two systems. The algorithm clusters on an absolute value towards zero, so it is only a matter of the user choosing which path to take, minimize cost through clustering large values in a positive matrix, or maximizing savings (reducing cost) through clustering larger numbers in a negative matrix. In the following examples the positive version of the matrix was used to represent relationship strengths and the objective function was to minimize clustering costs, which in turn, minimizes monetary costs.

The DSM was then processed through the collapsing DSM algorithm where the clustering percentage constraint was implemented in a range of increments from 10 to 60%. The cluster size constraint was set at 5 systems, and all other constraints were held constant from the initial clustering run to the final iteration. Each run was repeated ten times and the key indicators, cost history, and the solution and cluster matrices were recorded. The results

of the 60 runs are summarized in Figure 10 with arbitrary cost values which could easily be in thousands, millions, or billions of dollars.

The results presented in Figure 10 illustrate that if an enterprise chooses to integrate drastically in a short amount of time, it can expect to pay steep costs, but terminate after only a few periods of time. Periods of time are defined by the user and are represented by iterations of the algorithm, but will be referred to in this case as years. Conversely, an enterprise can choose to integrate slowly over time with less integration costs per year, but at the cost of time, in this case it will take the enterprise 21 years, instead of 5, to reduce the number of systems from 50 to 1. The underlying assumptions of course, being that the systems are able to be integrated or subsumed, that the process is completed before the next iteration begins, and that iterations are set at fixed intervals of time. However, as enterprises are concerned not only with annual costs, but with overall costs, Figure 10 illustrates the total cost of the integration plan (given the cost estimates provided at the creation of the original DSM are accurate).

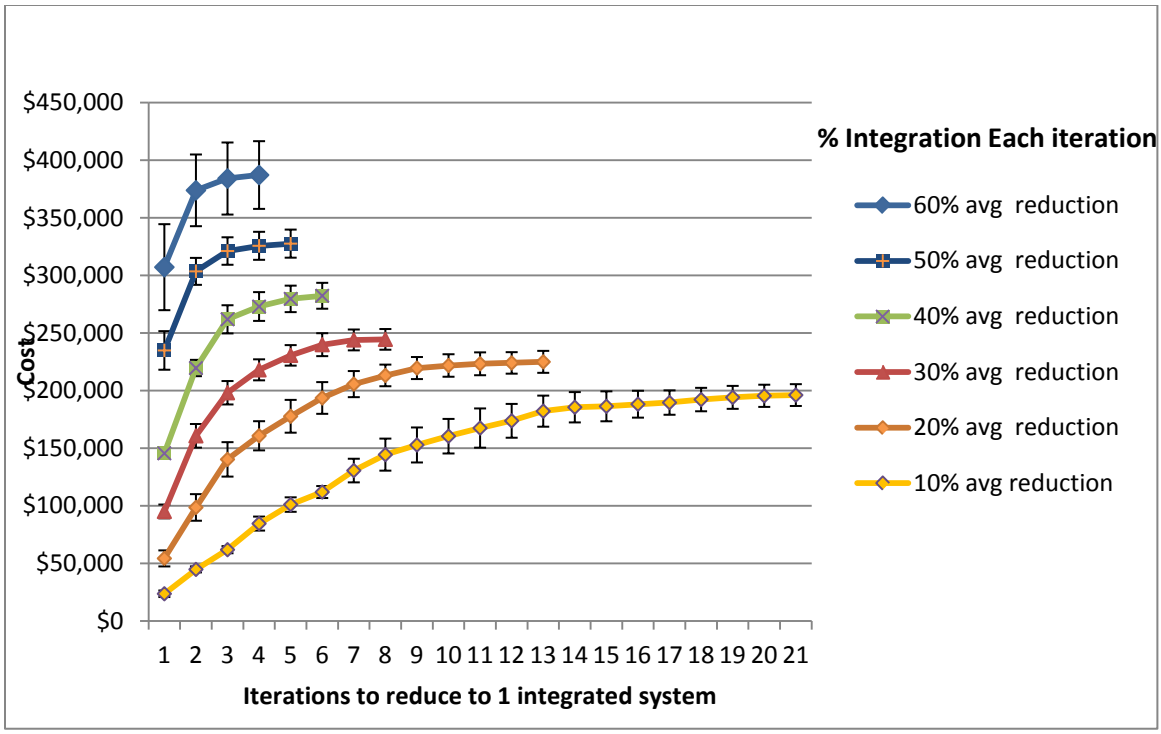


Figure 10. Cumulative Integration Cost by Iteration, varying targeted % reductions

In Figure 10, the total cost represented by the last entry of each curve provides some insight as to where the enterprise can expect to pay the least integration costs overall. In this example, keeping a steady 10% reduction each year provides the least overall integration cost solution, but does not tell the whole story. The curvature of the lines can be attributed to the fixed percentage reduction of the number of systems integrated per iteration and the associated costs (e.g. 100 to 50 systems, then 50 to 25, then 25 to 13, etc.). Again, in an application setting, it is doubtful that a user will continue to run the model at the same rate without making adjustments to better serve their enterprise, but this example illustrates the method without user induced variability.

There are other costs not currently covered in the model, such as Operational and Maintenance (O&M) costs, that could provide more savings if reduced earlier in the plan.

Integrating rapidly will induce higher integration costs, but those could be offset by reduced O&M costs are realized every year a system is turned off. Clearly, there are more factors that are important to an enterprise to make these decisions and should be calculated. This is a shortcoming of the model that will be addressed in future research.

Discussion and Conclusion

The algorithm was able to produce iterative integration plans for DSMs of various sizes, but was naturally dependent on the user understanding the feasibility of the solution space when constructing the constraints. If the percentage of matrix reduction was set to too high and the maximum the number of systems to integrate per iteration was too low, the solution space was over-constrained. DSMs with positive values demonstrate the clustering on element similarity, while DSMs with negative relationships demonstrate clustering on integration costs. Traditional methods have been successfully used to cluster a DSM, but none have been used iteratively. This collapsing DSM algorithm allows for researching potential future states of an enterprise and a path to reach those future states. Cost, in the monetary sense, is rarely modeled directly in DSMs. This method provides the framework for future economists as longitudinally focused algorithms that include cost will invariably require buy/integrate-now or buy/integrate later decisions where net present value (NPV) should be incorporated into the collapsing algorithm. If NPV is incorporated it will need to account for non-uniform integration lengths as integrating only two systems in one iteration may not take one year whereas integrating fifty systems will most likely take much longer than one year. Future research on cost could shed some light on whether or not there are

patterns or generalizable distributions that represent integration costs well, and can be used in random number generation.

There are many factors that can go into the values for each cell in the original DSM that represents the IT systems of an enterprise. As the number of factors was exhaustive, integration cost was chosen as the single factor to illustrate the usefulness of this algorithm, however future work will investigate which factors are most influential in industry making decisions and try to incorporate them using advanced decision analysis techniques that employ normalized weighting criteria provided by subject matter experts to generate values for DSM entries. More future work will investigate an Enterprise Resource Planning (ERP) case study on either a Government or private sector effort to validate the algorithm's performance and validity in a real-world environment.

V. Using Collapsing Design Structure Matrices to Develop Enterprise Systems Integration Plans and Cost Estimates

Abstract—Product or system-level research has dominated the study of traditional Design Structure Matrices (DSMs) with minimal coverage on enterprise-level issues. This paper utilizes a recently introduced method of collapsing DSMs (C-DSMs) to illustrate and mitigate the problem of enterprise information system (IS) redundancy while developing a systems integration plan. Through the use of iterative user constraints and controls, the C-DSM method employs an algorithmic and unbiased approach that automates the creation of a systems integration plan that provides not only a roadmap for complexity reduction, but also cost estimates for milestone evaluation.

Index Terms—System analysis and design, enterprise resource planning, interconnected systems, engineering management, design structure matrices, reduced order systems

Managerial Relevance Statement—This research paper provides an illustrative example of a large complex enterprise, riddled with information technology (IT) redundancy, that must find a way to reduce costs through elimination of redundant efforts. This example starts by building a relational matrix made up of several potential IT factors that use relationships between systems to construct cost estimates. Once the relational and cost factor matrices are constructed, user constraints are introduced and the C-DSM algorithm is enacted. Managers familiar with systems engineering and DSMs will benefit from the concept that DSMs can be re-scoped for the purpose of identifying redundancy in their enterprises and then utilized to produce a transition plan and implementation cost estimates.

Introduction

Since the introduction of factory production lines, corporations have spent enormous effort in the optimization of individual tasks resulting in complex hierarchal management schemes. After the introduction of computers as a tool for further optimization, the problem of sub-optimization continued to grow albeit unforeseen at the time. Through decades of competition, troubled corporations have failed, merged, or were hostilely taken over resulting in larger more complex corporations. As these corporations grew into vast enterprises before the Information Age, sub-optimization and redundancy of effort flourished and these hidden costs troubled CEOs in their efforts to maintain increasingly large enterprises without the technology to share information. Once networked databases came online and an enterprise view of information systems came into focus, the problem of disjointed, heavily duplicated, and incompatible resources became apparent.

Fu et al. described these problems and suggests some effective solutions (Fu Xiang-ling et al., 2010). A century of task separation, industrial process improvement, and departmentalization, combined with the explosion of automated business solutions, has resulted in "islands of information." Within these islands of information, separate systems were collecting similar data (frequently with inconsistent data types, naming conventions, etc.), but could not share the data efficiently, resulting in the need to build translators between these islands. A few ways to approach this problem include: starting with a new unified data system, build a unified "nervous system" to integrate the islands, or integrate groupings of systems that may cross departments (Fu Xiang-ling et al., 2010).

To optimize for the sake of the enterprise and to reduce redundancy, many enterprises sought out self-improvement efforts such as Business Process Reengineering (BPR), Total Quality Management (TQM), Lean, Six-Sigma, or implementing an ERP system. Success with these techniques has varied from complete failure to lackluster return on investment (Umble, 2003). IBM was successful in avoiding an enterprise breakup in the 1990s through invested leadership and enterprise-wide transformation by divesting itself of wasted effort and redundancy using BPR techniques (Gerstner, 2002; Lefever et al., 2011). Government enterprises such as the Department of Defense (DoD) and the Department of Homeland Security (DHS) also have this problem as reported by the Government Accountability Office (GAO) in 2007. The DHS has over 500 financial systems that have been identified for integration into a single system (U.S. Government Accountability Office, 2007) and the U.S. Air Force's Expeditionary Combat Support System (ECSS), which was trying to replace approximately 250 core logistics systems with a single Enterprise Resource Planning (ERP) solution, was cancelled in 2012 after over $1 billion was spent (Reilly).

Enterprises need supportive tools to generate transition plans that allow them to see both their current state (the "As-Is"), and their desired state (the "To-Be"). "Companies require a realistic route to implementation that sequences migration and places the services within a coherent organization design that ensures rigorous, effective governance." (Roghe et al., 2013) There are supportive methods for developing As-Is and To-Be architectures, using various architecture frameworks, such as the Zachman framework (Zachman, 2010), however, there is a gap in the literature explaining transition planning.

Background

Literature Review of DSMs

The basic design structure matrix (DSM) models systems for analysis and integration. DSMweb.org defines DSM as "a simple tool to perform both the analysis and the management of complex systems. It enables the user to model, visualize, and analyze the dependencies among the entities of any system and derive suggestions for the improvement or synthesis of a system." The purpose of the matrix is to explain the relationship between the system, or systems-of-systems, under study (see Figure 11). Most importantly for the collapsing feature of C-DSMs is a DSMs ability to allow for mathematical manipulation of the relationships.

System	1	2	3	4
1	1	0	0	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1

System	1	2	3	4
1	1	0	0	.5
2	0	1	0	0
3	0	.75	1	0
4	.25	0	0	1

Figure 11. Example DSM capturing relationships (left) or strength of relations (right) between components and systems

As DSMs have been used for multiple applications, rows and columns can represent any number of characteristics such as system architecture, organizational/team-based DSMs or activity/process DSMS (Browning, 2001b). A potential taxonomy for legacy enterprise information systems integration is presented in Table 10.

Table 10. Potential Taxonomy for Enterprise IS DSM Relationships

Type	Example Dependencies
Structural	▪ Number of Interfaces/Interface control documents (ICDs) ▪ Interface complexity/scope (Point-to-point interface (P2P), Enterprise Application Interface (EAI), batch/ real-time, loose coupling, service encapsulation, standard implementation) ▪ Projected Lifespan (how long is it required) ▪ Maintainability, Adaptability, Flexibility, Amount of change
Functional	▪ Shared functionality
Informational	▪ Number of information exchanges (in/out) ▪ Frequency of exchange (daily, real-time, monthly) ▪ Diversity of exchanges (transaction types, batch) ▪ Volume of data across the interface ▪ Common data elements ▪ Interoperability
Implementational	▪ Commonality of Programming language ▪ Likelihood of successful integration ▪ Performance requirements (Service level agreements)
Financial	▪ Cost to integrate/modify ▪ Cost/schedule to translate data

Central to the C-DSM method is the use of static DSMs as described by Browning (Browning, 2001b). Browning's seminal article provided history and a basis for all subsequent DSM research since its publication.

Extensions to Traditional DSMs

Although Browning had summarized and established a centralized basis for DSMs, several researchers have expanded DSMs into methods beyond the traditional DSM. Steven Eppinger has published several works on the extension of DSMs (Pimmler & Eppinger, 1994) (Sosa et al., 2003) covering the many different forms of DSMs such as Multiple Domain Matrices (MDM) and Domain Mapping Matrices (DMM). Helmer et al. continued Eppinger's work and included clustering methods and the use of five-dimensions per matrix

element (Helmer et al., 2010) and others . Chen and Huang then took DSMs to the supply chain community which provided a macro-level DSM example that was not focused on product design (Shi-Jie Chen & Huang, 2007). Other DSM method extending research can be found in these works: (Bartolomei et al., 2012; Brown et al., 2011; Browning, 2001b; Chang et al., 2011; Y. Chen et al., 2010; Dianting et al., 2013; Guenov & Barker, 2005; B. Hamraz et al., 2013a; Helmer et al., 2010; Holley et al., 2014; Kasperek et al., 2014; Lagerstrom et al., 2013; Lambe & Martins, 2012; Lancaster & Cheng, 2008; Liang, 2009; Mikaelian et al., 2012; Nakata, 2009; Sharman & Yassine, 2007; van Beek et al., 2010).

Gaps

The problem of enterprise integration has been examined and of particular note is the complexity of the relationships between legacy systems. The authors examined the problem from a systems theory perspective and evaluated several graphical systems engineering methods to display complex systems for the purpose of understanding and CEO-driven systems integration. It was found that DSMs provided the most succinct graphical display while still providing a mathematical conduit for guiding a clustering routine with the eventual goal of generating automated transition plans. It was also found that the overwhelming majority of DSM research, about 86%, focused on product-level investigation. Of the remaining research only a slight percentage held a global or enterprise focus. Of particular note, a recent article from Lagerstrom et al. also noted this gap in stating, “we have yet to see [DSMs] deployed in enterprise architecture modeling” in their search for hidden structures in software architecture (Lagerstrom et al., 2013).

In previous work, the C-DSM method was presented in detailed explanation and is summarized in Section III of this paper. Although the authors were able to develop integration plans using the C-DSM method, the clustering routine was driven off of one factor (integration cost) and did not include any other cost estimates for detailed decision making.

The C-DSM Method

Collapsing DSMs

C-DSMs use constraints and macro-level iteration to collapse traditional DSMs by integrating clustered systems and making use of the diagonal to ensure relationships are not lost through the collapsing process. Traditional DSMs describe the relationships between elements in each matrix entry, but leave the diagonal (e.g. matrix entry 1,1; 2,2; 3,3; etc.) unused as the relationship between a system and itself is irrelevant. The C-DSM method utilizes the diagonal to store algorithmic results from previous iterations as an integrity check to ensure the collapsed DSM did not lose any of the original DSM's relationships. The use of iteration and constraints will be described further in this section while preliminary definitions are provided in Table 11.

Table 11. Collapsing DSM definitions

System X_i: one of the $i=1..n$ Enterprise systems
x_{ij}: relationship, strength or relation, or cost of integration between systems i and j
Clusters k: Clusters represent a group of highly related systems that are "most alike" which indicates higher likelihood of compatibility for the purpose in integration. With greater compatibility there should be less difficulty and cost in the integration
Total Coordination Cost: Objective function that quantifies the cost of clustering systems, penalizing for too much inter-cluster relations and cluster size

The objective of the C-DSM algorithm is to minimize the off-diagonal elements through clustering similar systems, integrating (or collapsing) those systems into single systems with the goal of providing a less complex DSM. Each repetition of the algorithm produces the original clustered DSM and two "collapsed" DSMs. The process repeats until the process owner has obtained their desired level of integration, or until the DSM has collapsed down to a single system. One repetition represents an arbitrary time period that is defined by the process owner and the speed of integration is controlled through user constraints and controls.

Assumptions

This research utilizes matrices with the assumption that there is some quantifiable measure of similarity between two systems. Similarity can be any measurement of the relationship between systems as indicated in the taxonomy in Table 10 or any other measurement defined by the process owner. It is believed, that a higher measure of similarity equals more redundancy between systems or at least more similarity in the functionality between systems. In previous work, the authors utilized integration costs as a single measure that would invariably represent any combination of factors down to its most basic determining factor for an enterprise: monetary cost.

Foundational Algorithm

As polynomial optimization problems with multiple constraints quickly become NP-hard it is appropriate to employ heuristics such as genetic algorithms, Tabu search, and simulated annealing (SA). Combining the problem of large enterprise redundancy with restrictions on cost and speed of integration, SA is most appropriate as it is well suited for

searching for global optimums in a fairly large, but defined, solution space. SA may not produce the "best" solution, but one of the tools of C-DSM is in repetition where constraints and decision variables are updated frequently to guide a path to improvement (not necessarily the "best") provided there is improvement to the objective function. In 2001, MIT student Ronnie Thebeau (Thebeau, 2001) created a series of MATLAB functions that articulated an SA algorithm developed by (Fernandez, 1998). The authors augmented the core functions of this MATLAB code to allow repetition for the purpose of producing an integration plan of optimal reductions. This original algorithm follows:

1. Each element is initially placed in its own cluster
2. Calculate the Total Coordination Cost of the Cluster Matrix
3. Randomly choose an element *i*
4. Calculate bid from all clusters for the selected element *i*
5. Randomly choose a number between 1 and *rand_bid* (algorithm parameter)
6. Calculate the Coordination Cost if the selected element becomes a member of the cluster with highest bid (use second highest bid if step 5 is equal to *rand_bid*)*
7. Randomly choose a number between 1 and r*and_accept* (algorithm parameter)
8. If new Coordination Cost is lower than the old coordination cost or the number chosen in step 7 is equal to *rand_accept*, make the change permanent otherwise make no changes
9. Go back to Step 3 until repeated a set number of times*

*Steps 6 and 8 enact simulated annealing by randomly accepting the second highest bid (6) and accepting changes even if the overall solution worsens (8) to avoid local optima entrapment.

Objective Function

The objective of the C-DSM algorithm is similar to the original algorithm: to minimize the absolute value of the Total Coordination Cost through re-clustering. Absolute value is used in this algorithm as some factors have a positive scale (similarity, number of interactions), and others have a negative scale (operations and maintenance costs, integration costs, etc.). The calculation for Total Coordination Cost follows in Equation 1.

$$Total\ Coord\ Cost = \sum_{k\,=\,1..z} \sum_{ij\,=\,1..n} \left(ExtraClusterCost_{ij_k} + IntraclusterCost_{ij_k}\right)$$

Equation 3

Where:

$$ExtraClusterCost_{ij_k} = \begin{cases} (x_{ij} + x_{ji}) * n^{\,powcc} & \text{if } c_{ik} \neq c_{jk} \\ 0, & \text{otherwise} \end{cases}$$

$$IntraClusterCost_{ij_k} = \begin{cases} (x_{ij} + x_{ji}) * ClusterSize_k{}^{\,powcc} & \text{if } c_{ik} = c_{jk} \\ 0, & \text{otherwise} \end{cases}$$

z = number of clusters in the solution
n = number of elements (number of systems in the matrix)

Constraints

The original algorithm was an unconstrained multiple objective formulation. The extensions for iterative use in element reduction (collapsing) resulted in adding user constraints on max cluster size τ, minimum number of clusters δ, and maximum number of clusters ε. These constraints prevent the algorithm from integrating beyond the capabilities or resources of the organization during each iteration. Typically, clustering routines in static DSMs allow elements to belong to more than one cluster. For example, if a driveshaft connects two different components the driveshaft would be the overlapping element in two clusters (Browning, 2001b). Given the goal of integration of large numbers of physical systems, multiple cluster memberships are not allowed. This departure from the original algorithm is enforced through an added constraint in the clustering function and solutions with multiple cluster memberships are rejected. The constraints are simply:

Table 12. Table of Constraints

$\sum_{l=1}^{m} c_{lk} \leq \tau$	$\sum_{k=1}^{p} c_{ik} \geq \delta$
$\sum_{k=1}^{p} c_{ik} \leq \varepsilon$	$\sum_{i=1}^{p} c_{ij} = 1$

Collapsing DSM Algorithm Integrity Check

Once Thebeau's algorithm produced a clustered DSM that met the user-defined constraints, a separate DSM is constructed to represent the next evolution of the DSM. This reduced DSM integrates clusters into single systems and the new DSM's size is equal to the previous solution's number of clusters. To ensure relationships and relationship strengths were not lost, the clustered DSM's algorithmic cost metric is compared to the reduced DSM's cost metric. The dimensions of the DSMs are different, but the metrics will be equal as the Extra-Cluster Costs are recorded in the off-diagonal elements and the Intra-Cluster Costs are recorded on the diagonal elements of the reduced DSM. Entries for the collapsed matrix are calculated based on average sum of the associated elements in the source DSM and then multiplied by a factor produced by the user (see Equation 2). The user should understand the relationship of the systems under consolidation to determine if the relationship of a combined system is stronger or weaker in relation to the other systems in the enterprise. Upgrading one cluster of systems with current technology should make it easier to integrate in the future, but there may be cases where the opposite is true. Allowing the user to control this factor allows for more user customization in modelling their enterprise.

$$\text{Collapsed Matrix Entry} = \frac{\sum_{ij=1}^{k} x_{ij} + x_{ji}}{k} * \lambda$$

Equation 4

Where:

λ = user defined multiplier for estimating the new C-DSM entry

Cost Calculations

There are several cost calculations built into the C-DSM method. The first clustering cost metric is used in the foundational algorithm to determine which clustering solution is the most optimal. Once cluster memberships are determined, the other costs are calculated in the collapsing algorithm. Integration costs represent a monetary value that represents the resources required to integrate clustered systems into one system, terminate redundant systems leaving one system, or replace all clustered systems with a new all-inclusive system (such as an ERP). Integration costs for the current iteration, as well as the entries for the integration cost matrix for the proceeding iteration are calculated. The proceeding integration cost and relationship matrices are constructed using the same formula as presented in Equation 1. O&M costs are finally calculated at this time following the same methodology and formula with the exception being that it is an array versus a matrix.

Now that the algorithm has been presented, an example case study follows using source data from a government project currently ongoing.

Case Study

Creating Source Matrices & Arrays

The following business case for demonstrating the C-DSM method is based on government data and cost estimates for a large enterprise solution currently in progress. Some information is masked, other values are inspired by real systems, and some random values are generated to represent a realistic solution based on the true estimates provided.

The core calculation that drives the C-DSM algorithm uses a matrix constructed that represents the pair-wise relationships between systems. This driving relational value can be constructed in any fashion, but must capture the relationship between systems and not just descriptive information about either system. Static traditional DSMs cluster to minimize coupling while increasing cohesion between highly related elements. The values placed in the matrix must represent the relationship between the two systems, but in C-DSMs a similarity or percentage of estimated likeliness can be used for crude estimations. In previous work, integration cost was presented as a stand-in representation for any and all factors.

For more precise and thoughtful investigation of the relationship between two systems a normalization table could be employed where all of the factors deemed useful can be evaluated and combined into one value. To provide more fidelity to these values, an enterprise CEO could weight the importance of the suggested factors and some advanced decision analysis techniques could be employed (Keeney, 1992), (Kirkwood, 1992), (Shoviak, 2001). It is also recommended that costs are evaluated and compared separately from other similarity factors. If using subjective criteria and scales there should be some

attention to inter-rater reliability to mitigate discordant ratings. See Table 13 for an example of possible factors. As direct data was not available concerning these factors a Relationship Matrix was generated randomly using a Chi-square distribution using two degrees of freedom to generate a non-uniform relationship workspace to simulate attribute values.

Table 13. Example DSM Entry Calculation Table

Attributes	**Value Measurement Scale**										**Value** *V(x)*	**Weight** *W*	**Weighted Score** *WV(x)*
X1. Percent of functional similarity	0-10%	20%	30%	**40%**	50%	60%	70%	80%	90%	100%	**0.4**	**0.5**	**0.20**
X2. Percent of data similarity	0-10%	**20%**	30%	40%	50%	60%	70%	80%	90%	100%	**0.2**	**0.1**	**0.02**
X3. Ease of implementation integration	Very Difficult		Difficult		Typical		Somewhat Easy		Easy		**0.1**	**0.3**	**0.3**
	1	2	3	4	5	6	7	8	9	10			
X4. Frequency of communication	Do not/ Very Rarely		Rarely		Sometimes		Frequently		All the time		**0.1**	**0.1**	**0.01**
	1	2	3	4	5	6	7	8	9	10			
Weighted Score for cell entry in the Relationship DSM (Sum of Weighted Scores) $\sum WV(x)$													**0.26**

Another method to produce a relationship DSM that would remove some of the influence of man-made decision and place it in the objective realm of machines would be to use an interoperability measurement in combination with some of the taxonomies listed in Table 10. Interoperability can be calculated by "measuring the interoperability of a heterogeneous set of systems, experiencing any type and number of interoperations" as illustrated in (Ford, Colombi, Jacques, & Graham, 2009a; Ford, Colombi, Jacques, & Graham, 2009b). For a practical application using supply chains see (Johnson, Ogden, & and Ford, 2010)

To produce cost estimates resulting from any proposed changes the authors obtained draft planning documents for an ERP which contained O&M estimates. A 100 system O&M array with random numbers was generated where the overall average cost was $4.79M, a minimum of $0.02M and a max of $12.65M similar to the range of values from the government data in Table 14. To generate a representative Cost Integration Matrix a 100x100 matrix was created from random numbers with a 3-30 range representing the average integration cost per system estimate in Table 14.

Table 14. Cost Source Data

System	Total Cost*	O&M New Sys*	Integration Costs*	O&M % of integration cost	Num Legacy Sys	Avg Int Cost Per Sys*	Avg O&M Old Sys*	Total O&M Old Systems
System A	137.5	4.5	124.8	3%	17	7.34	1	17
System B	86.4	7.8	70.2	9%	18	3.90	1	18
System C	83.9	4.8	68.7	6%	22	3.12	1	22
System D	159.7	7.9	132.4	5%	11	12.04	1	11
System E	32.2	0.7	29.3	2%	1	29.30	1	1
System F	18.9	1.2	16.6	6%	1	16.60	1	1
System G	44	6	25.6	14%	6	4.27	1	6
Total / Avg	562.60	4.70	467.60	6%	76	10.94	1	10.86

*Cost in $Millions
Old system O&M costs not yet available and were estimated to be equal to avoid undo influence

Defining User Constraints

The novelty of the C-DSM method is the iterative collapsing of a DSM to provide an integration plan, and one of its key strengths is its customizability for user defined application. To reduce the number of solutions provided, the user is given a set of constraints to define the limits of the solution space. The first constraint is the overall objective for the current iteration; the overall desired reduction percentage. For this descriptive case a 10% goal is set with an acceptable range +/- 2% to reduce 100 systems down to 88-92 systems. A

range is provided to allow the algorithm to explore answers that are slightly above and below the intended target as a "better" solution may be available. The algorithm rounds to the nearest maximum and minimum number of systems which results in discrete boundaries from continuous input (e.g. a 22% reduction of 33 systems must translate to a specific number of systems and not part of a system). The second constraint is the maximum number of systems to integrate per cluster. Given only the first constraint, the algorithm can produce solutions where all of the systems chosen to integrate could be in the same cluster meaning that an integration team would be given the difficult task of integrating ten systems in one integration period versus integrating five sets of two systems in one integration period.

The user is also asked to provide direction for the next step of the iterative plan through the use of three multipliers to assist in the estimation of the future costs of the integrated systems. These entries are used to calculate the next iteration's Relationship and Integration Cost matrices as well as the new O&M array. For example, systems A and B are combined to form system AB. If the relationship multiplier is set to 1, the new relationship entries for the Relationship Matrix for system AB are simply the average of the old entries of A and B. If set above 1, there will be a stronger resultant relationship with other systems in the enterprise. Setting this value to less than 1 would result in a weaker relationship between system AB and the other systems in the enterprise. This same approach is used for both the integration cost and O&M cost multipliers and the user has to estimate what the effects of integrating systems A and B will have on the rest of the enterprise (see Equation 3). Will their upgraded integration allow for easier connections to other systems in the enterprise? Will O&M costs of the new system AB be similar to the old costs of A and B? And finally,

will it be easier (and less costly) to integrate system AB versus separate systems A and B? The answer to these questions naturally lies in what systems A and B are. If they are dated obsolete legacy systems that require expensive support then an updated union of systems with updated programming should have less O&M and integration costs.

$$\textbf{Collapsed Matrix Entry} = \frac{\sum_{ij=1}^{k} x_{ij} + x_{ji}}{k} * \lambda$$

Equation 5

C-DSM Implementation

Once the constraints have been defined then the C-DSM can calculate the first iteration of the transition plan. As shown in Figure 12, the Excel spreadsheets are fed into the algorithm, followed by the user constraints and controls. The core modified Thebeau code calculates a series of acceptable solutions and returns the "best" solution. The solution exports the key solution indicators and matrices into another set of excel spreadsheets used to prepare the next iteration. A list of clustered systems are identified for integration and presented to the user. The user can then repeat this cycle until the number of systems reduces to a single system which provides a long term transition plan. However, in true application, once it is time to enact the next iteration, all of the relationships and costs should be reviewed for currency given the successes and failures of the previous iteration's plan. Market conditions and lessons learned from the first integration period will most likely provide improved and timely information for a more accurate and relevant solution for the next iteration. It is not necessary to continue until one system remains as it is typical for integration efforts to be accompanied by persistent legacy systems that cannot connect to

other systems or be replaced for legal or other reasons. Figure 12 shows the overall flow of the algorithm.

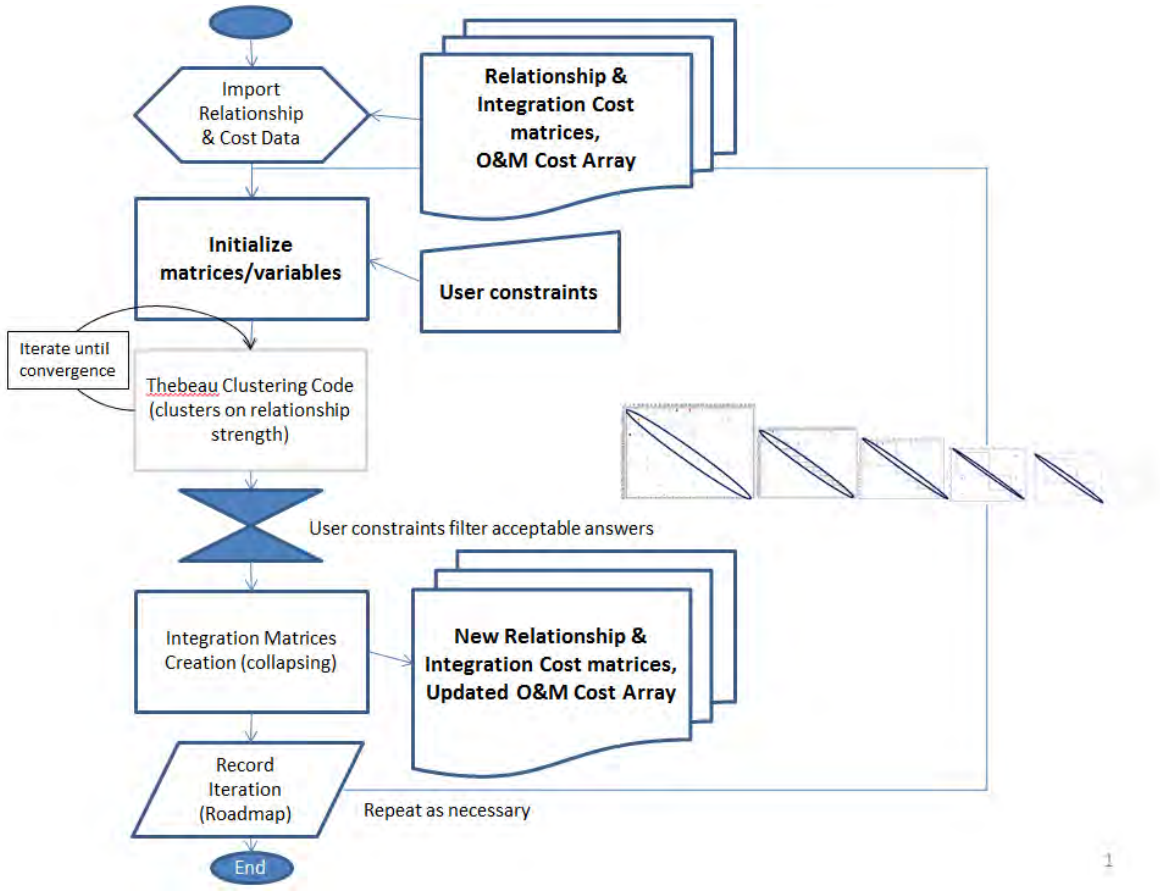


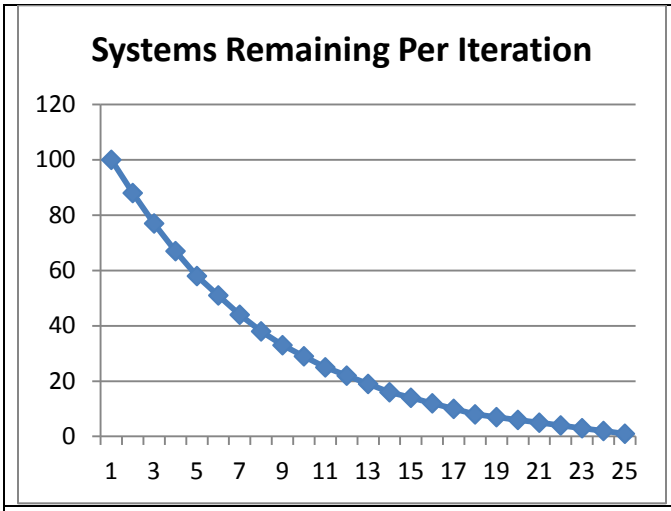
Figure 12. C-DSM Algorithmic Flowchart

Results

Resultant Transition Plan

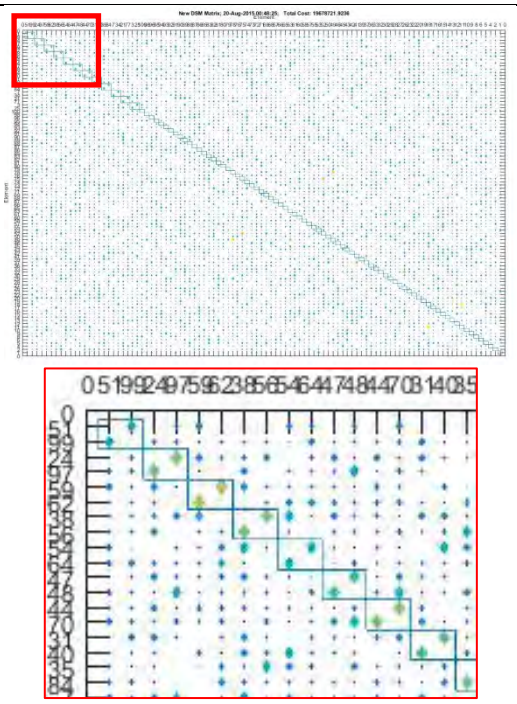
The final solution presented a 24 iteration transition plan that held to the static constraint of a ten percent (overall reduction of systems (see Table 15).

Table 15. Example Integration Plan



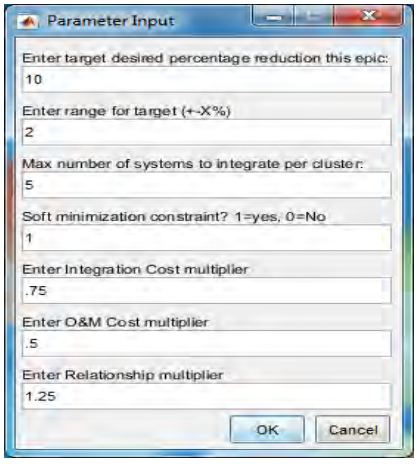
Iteration	Systems Clustered
1	43-10; 75-85; 71-80; 57-81; 54-64; 28-49; 47-66; 13-42; 17-96; 16-82
2	43-10; 75-85; 71-80; 57-81; 54-64; 28-49; 47-66; 13-42; 17-96; 16-82
…	…
24	1-2

Iteration	Number of Systems	Percent Reduction Achieved	Integration Cost this Iteration	O&M Savings this Iteration
1	88	12%	$498M	$59.18M
2	77	12.5%	$293M	$62.02M
…	…	…	…	…
27	1	50%	$0.04M	$5.42M
Total Cumulative Cost of Solution: $1.35B			Total Cumulative Savings of Solution: $37B	



Iteration 1

In the first iteration, it can be seen that the larger diamonds, which represent a stronger relationship, are pulled into clusteres with other systems where they are strongly connected or functionaly similar based on the user inputs of the relationship DSM constructed in Table 15.



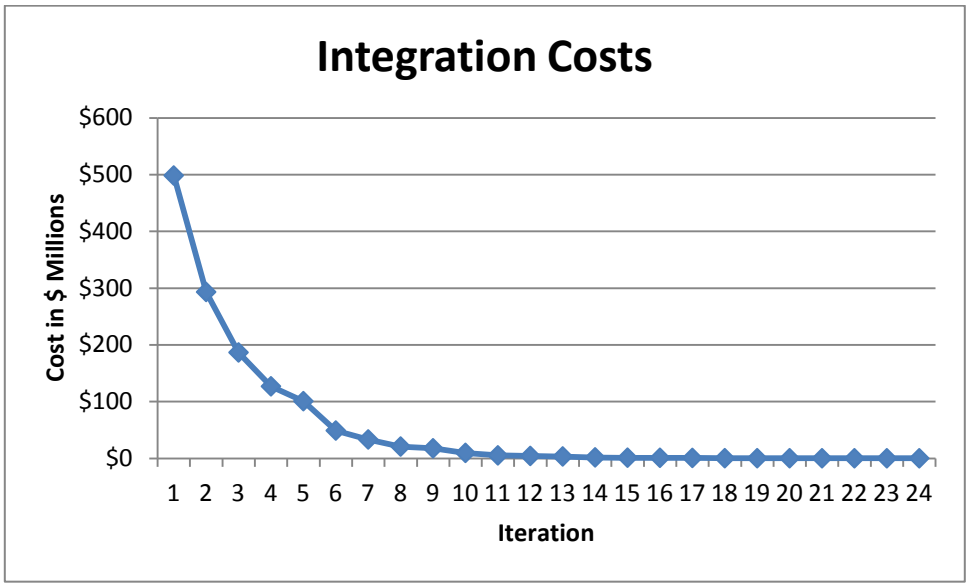
Cost Estimates

The following cost estimates are based on randomly populated matrices representing the data provided in Table 14, and are the result of 24 iterations where the user did not modify or update any cost estimates or alter the rate of integration through constraint manipulation as is intended in a true application sense. This example demonstrates the abilities of the C-DSM method without user influence under the multiplier conditions hypothesized where future systems are more strongly related, and integration and maintenance costs are less. The parameters for this example are as follows: 10+-2% overall reduction with a soft minimum clustering constraint, and a 5 system maximum cluster size. The cost multipliers are: Integration Cost Multiplier = 0.75; O&M Cost Multiplier=0.5; and Relationship Multiplier=1.25. The average percent reduction for the first 15 iterations was 11.14%, but once the number of systems in the enterprise fell to less than 15 systems it became mathematically improbable and eventually impossible to reduce two systems and only yield a less than ten percent reduction of systems. In the event where the constraints create impossible conditions the algorithm will present the best solution found along with an error indicating that the solution does not meet all of the user constraints and suggests that the user relaxes a constraint and rerun the model. In this case, all final presented solutions were accepted as they were understood to be as close to the user's intent as mathematically possible.

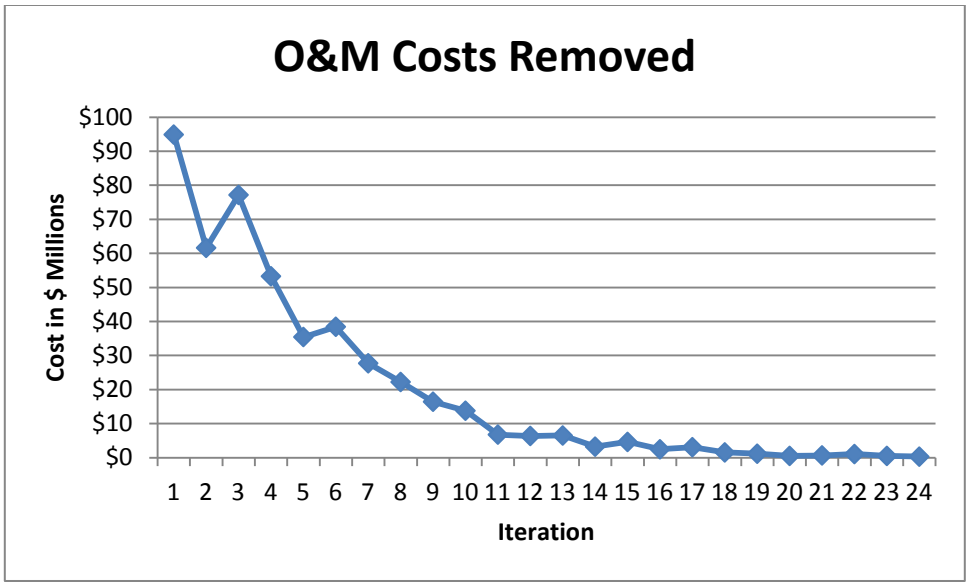
The total integration cost for this solution was $1.35 Billion over 24 integration periods with an average of $56.41M per year. However, as can be seen from Figure 13a, using the average cost per year would not useful for planning purposes due to the volatility of

the curve. The jagged increases are due to mathematical differences between continuous percentage-based constraints and the algorithm's rounding to the next highest discrete number of systems (e.g. sometimes when the algorithm rounds up, it includes more than the percentage constraint, but still holds the discrete constraint equivalent in the current iteration).

The total O&M costs removed for this solution was $ 479 Million over 24 integration periods. This would result in complete removal of O&M costs so it can be determined that the O&M multiplier is not appropriately adjusted, at least for the case of leaving it constant, for the full 24 iterations. However, O&M savings are earned through cost avoidance per iteration following integration resulting in a 24 integration period savings of $5.04 Billion (Figure 13b). The original enterprise annual O&M costs were $479M while the final system solution's O&M costs are $1.81M annually. As systems are clustered based on some measure of relationship strength, and not directly monetary cost, the inexpensive "low hanging fruit" are not selected first unless they are highly related to other systems. This may run counter to some decision makers as it is common (as it is easy) to address those systems with "gut decisions," but may explain the overall downward trend of Figure 13a. It is understandable to believe that integration will continue to get more expensive as more complex and dis-similar systems remain after several iterations, but this is where the cost multipliers add value to this model. It is hypothesized that highly disparate systems will cluster away from each other, but over time and upgrade will become less disparate as they integrate with other, more alike systems. The integration estimation multiplier can aid decision makers in producing estimates as they progress through iterations.



(a)



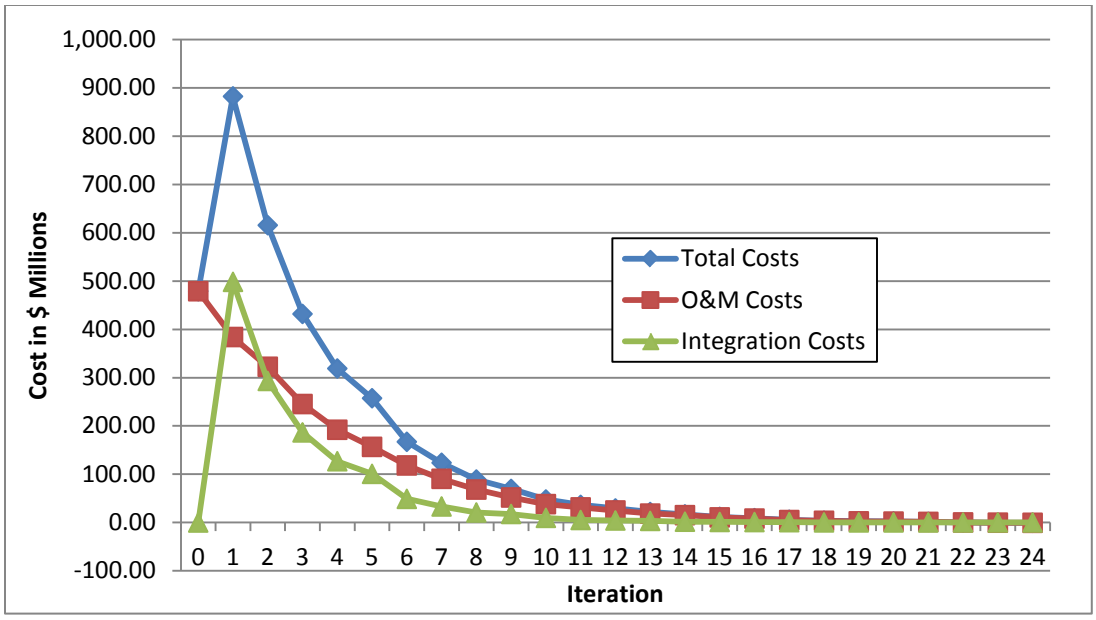
(b)

Figure 13. Example Cost Estimates Results

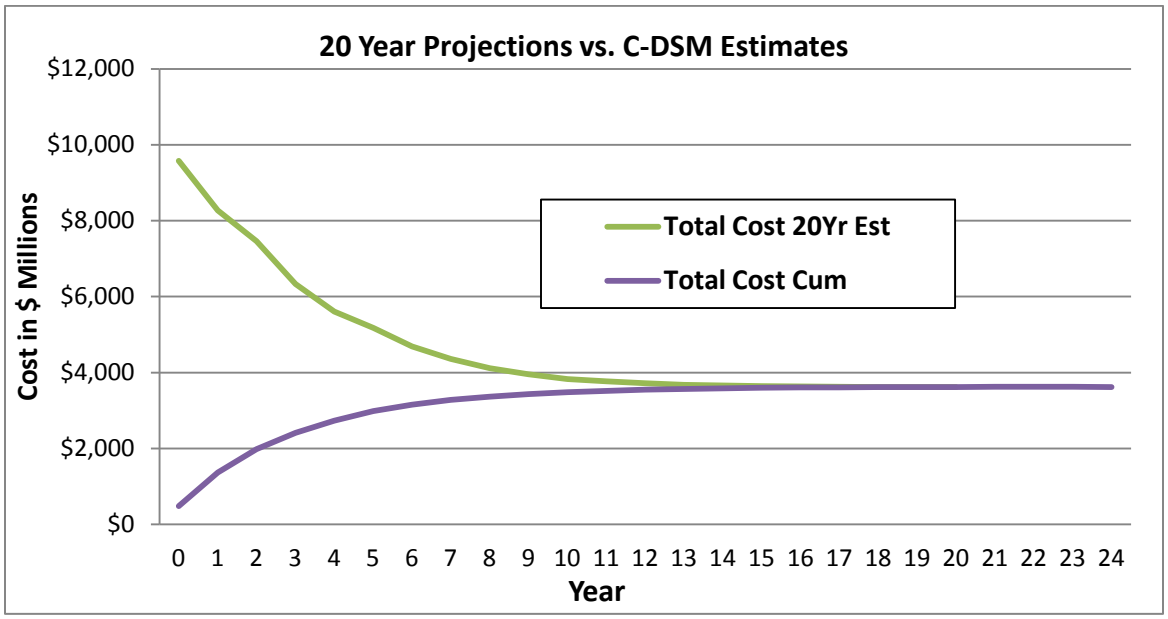
Total costs were then graphed to provide decision makers a few points to consider. In Figure 14a, the baseline at time zero represents the current configuration of the enterprise.

Then, at iteration 1, integration costs are spent and O&M costs are removed resulting in the total cost of the solution if the decision maker decides not to integrate further. However, as the iterations continue, a downward trend is realized after iteration 3 where the total cost drops below the baseline O&M cost. After three iterations, the case study enterprise has begun to see the financial benefit of integrating from 100 to 67 systems. This chart illustrates the benefit of the C-DSM approach as planners can choose to stop integrating at any point and do not have to "buy-in" to a full ERP (many-to-one integration plan), but can instead integrate to their desired amount of integration.

Persistent legacy systems are most likely to be involved in an integration effort concerning information systems as some may not be replaceable or may be required for archival or legal reasons. These can be accounted for in the model by making their entries vastly different from the other elements (negative vs. positive values) and represent a case where reducing the enterprise to one system is not possible and is yet another reason the C-DSM method can provide a useful planning horizon for decision makers.



(a)



(b)

Figure 14. Total Cost Estimates and Projections

In Figure 14b, a 20 year projection is calculated to determine what the total cost of the solution would be at the 20 year mark starting from Year 1. This table also starts at the baseline of the current enterprise configuration and provides the total cumulative cost should

the enterprise chose not to integrate. At Year 1, given iterations are equal to years, the cost is calculated by calculating 19 years of O&M savings and subtracting off the one time cost of integration. At Year 2, the cost is calculated by the first year of O&M savings, plus 18 years of the new O&M savings, minus the sum of the two years of integration costs. This trend continues for 20 iterations and produces the descending cost projection curve in 15b while the increasing curve represents the actual yearly total cumulative cost calculated by the C-DSM method.

Sensitivity Analysis

The example in Figure 14 uses the integration cost multiplier of 0.75 based on a hypothesis that the future cost of integration with current programming will be less than the initial cost to integrate software that is several decades old. Also, the O&M multiplier of 0.5 was also chosen under the hypotheses that future systems would be considerably less costly than current software. And finally, a relationship multiplier of 1.25 was based on the hypothesis that a future system designed out of integration with plans for future integration would have a stronger relationship with other systems, integrated in parallel. As these numbers are purely hypothetical and based on assumptions it was necessary to examine a few other combinations of these multipliers to get a better understanding of the C-DSM calculations.

The first alternative model set the relationship, cost, and O&M multipliers to 1 to illustrate the model using averages in essence removing the multipliers to gather a baseline in which to compare the other examples. The baseline cost results are shown in Figure 15. In this example, reduced O&M costs do not overtake the cost of integration until iteration 6 when the number of systems has reduced to 44.

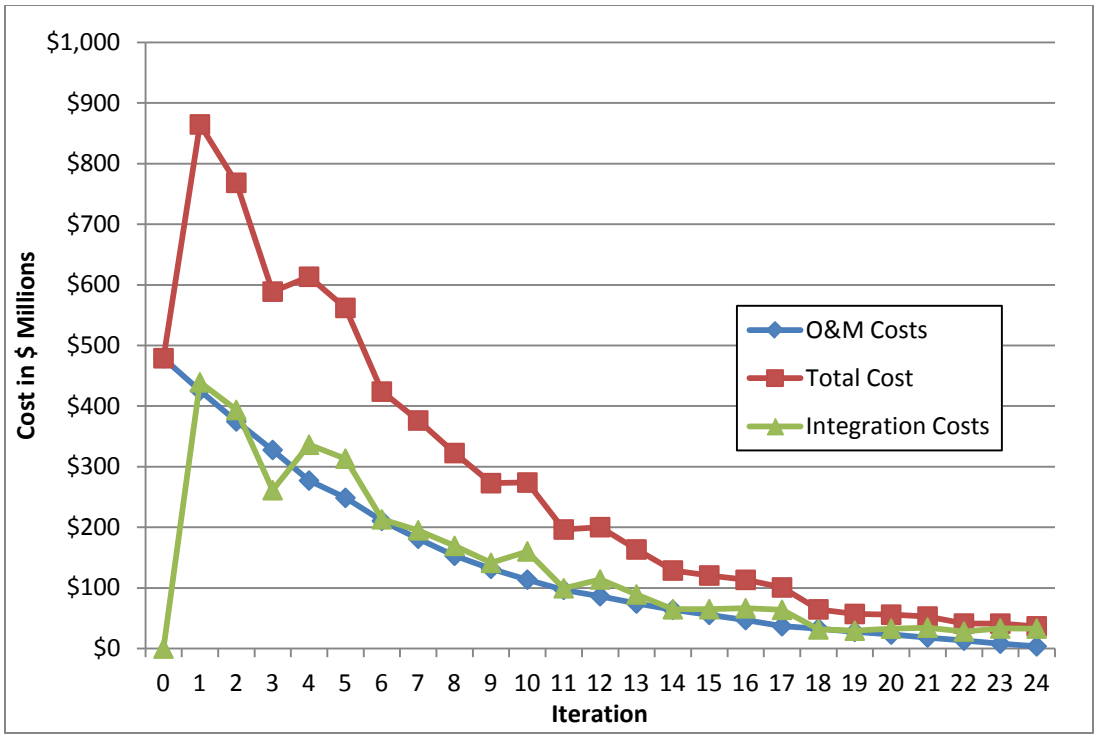


Figure 15. Baseline Model Cost Estimates

The second alternative model increased the cost multipliers to 1.5 and decreased the relationship multiplier to 0.5, essentially exploring the alternative that our hypotheses are not only incorrect, but in the opposite direction. All other constraints and variables remained the same. The costs are shown in Figure 16 and indicate that if costs rise as systems integrate the total cost of the solution will continue to rise without any savings in sight.

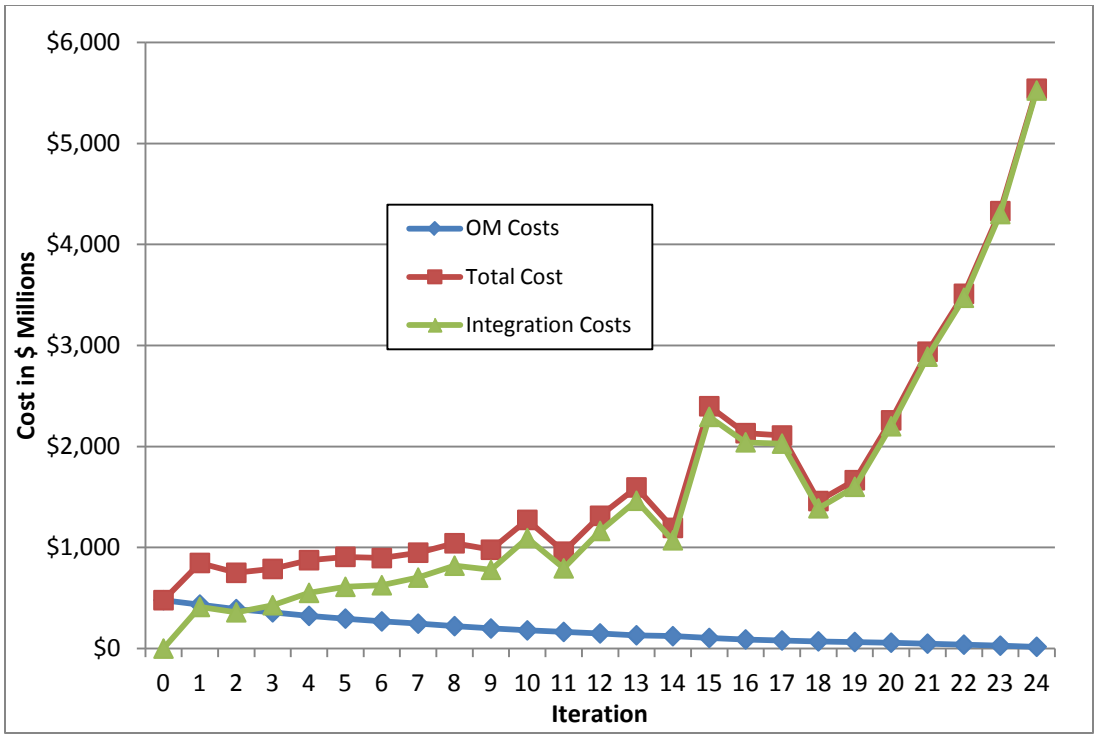


Figure 16. Alternative Model with 1.25 Cost Multipliers

These two alternative examples provide insight into the amount of control the multipliers have on the solution if not left at 1. In a practical setting, it will be imperative to compare estimates between iterations to grasp the proper multiplier setting.

Discussion and Conclusion

Discussion and Implications

Although the case presented in this paper is treated academically through strict control of the constraints, the utility for practitioners of the C-DSM method is apparent. Practitioners will be able to make adjustments to the constraints and controls to produce more applicable and more cost effective solutions. As the iterations proceed, practitioners will learn from the integration process and improve the information going into the source files,

which will in turn, improve the next iteration. This method provides an impartial long-term macro-level integration plan that is not dependent on a single process owner. It is possible for each iteration to be championed by the current CEO or enterprise integration officer and then passed on to the successor without need for drastic changes in methods once leadership changes occur.

Controlled by a systems owner the results should cost less and the integration should be faster as it is unlikely for an enterprise to continue on a set percentage for the complete plan without speeding up or slowing down the rate of integration. The source documents from the government system used in our example displayed a six year plan to integrate a vast number of systems and after similar integration efforts such as ECSS, GCSS, and DEAMS it is unlikely that such a plan will not change to extend to a much longer planning horizon. Given this assumption, the cost estimates from Table 14 will also most likely change and the C-DSM will have to be re-run to provide the process owner improved estimates.

Conclusion

In the corporate sector, one common practice is to select a single ERP and employ cutover strategies for all legacy systems; a many-to-one transition strategy. However, this C-DSM method could aid large enterprises that need to group disparate systems into larger conglomerate sub-systems prior to connecting to the primary ERP or other communications backbone.

The government sector will find even more utility in this method as change in government systems is much slower and the size of the problem is much greater.

Government evolution is stymied by politics, policies, laws, culture, a vertical (and sometimes overlapping) hierarchal structure, as well as other hindrances. There seems to be a much greater need for this method in the government sector and this method will allow long term planning that is independent from many of these factors while it can incorporate some of these factors in the generation of the source matrices and arrays along with setting the constraints and controls.

Limitations and Future Research

Method Limitations

There are some limitations to the C-DSM method that must be discussed before choosing to employ this method. One limitation of this method is the reliance on the source data. The old adage "garbage-in garbage-out" is especially true in this method and the relationship matrix drives the entire clustering algorithm. If an organization does not take the time to map the current state of their enterprise and answer those questions (or their own relationship describing questions as prescribed in Table 13 for each pair of systems), then the resultant solutions presented will not make sense. Depending on the size of the enterprise in question, this stage could take considerable time and resources. Additionally, if the integration cost estimates are highly inaccurate then the resultant calculations will also be inaccurate however they will not affect the transition plan as they are not included in the clustering decisions.

Future Research

As this method has just been developed and there is little research in the enterprise application of DSMs, this area is suitable for future research. The C-DSM in this paper is clustering on a single matrix that is populated using weighted normalized data, but there should be a way to combine separate matrices in the clustering decision. Cost is the bottom line in many business decisions, and has been used in previous work by the authors for clustering decisions, but the correlation between cost and relationship strength should be significant enough to cluster both of these factors at the same time. The authors plan to continue this work and develop a clustering method that combines a series of separate matrices for clustering decisions to answer this need. Other work includes more robust economic calculations using net present value (NPV) to assist in the integrate-now versus integrate-later decision. This method employs a long term plan that could be improved by incorporating actual yearly time increments and annual interest rates to improve cost calculations. The field of DSM research is growing and there are undoubtedly more ways to incorporate other's techniques with collapsing DSMs.

VI. Conclusions and Recommendations

Chapter Overview

The purpose of this chapter is to present the comprehensive conclusions and recommendations from the complete body of research presented in Chapters I-V. This chapter will begin by drawing conclusions from the entirety of this dissertation research and will address the research objectives and research questions. This chapter will then discuss the significance of this research and pontificate the benefits of the C-DSM method and the potential it has for USAF challenges. After the research significance has been presented, recommendations for USAF actions will be specified as well as recommendations for future research. This chapter concludes with a summary of this dissertation effort.

Research Conclusions

In reviewing the research presented in this dissertation, several conclusions can be made. It is clear enterprises can be modeled using DSM methods and that these methods are suitable for not only product or low system level application, but also system-of-system and high level application. In Chapters I and II, the problem was presented where the history of business evolution from the industrial revolution to the information age was examined to provide context for the growth of systems redundancy within vastly complex large enterprises. This problem was examined through the lens of several theories to provide a basis for determining the existence and strengths and weaknesses of available tools. It was found that systems, networking, and complexity theories provided the right sandbox for tool selection in attempting to address the USAF's problem of multiple redundant legacy information systems. In Chapter III, a paper submitted to a peer reviewed journal article, it

was found that the tool of choice was DSMs, and presented a literature review on DSMs in the context of enterprise-level literature. Gaps in the literature were found and presented an opportunity for this dissertation. Specifically, there was little research done with DSMs that took on an enterprise focus and their use with ERPs and BPR was virtually non-existent. As so eloquently stated by (Lagerstrom et al., 2013),"we have yet to see [DSMs] deployed in enterprise architecture modelling." Chapter IV was also submitted to a peer reviewed journal and represents the introduction of the C-DSM methodology. Chapter IV illustrates how DSM clustering and reduction can be iterated through the use of user constraints and allows for the conclusion that the C-DSM method can be applicable for enterprise integration. Finally, Chapter V presents a case study that capitalizes on government data, illustrates a method to develop source DSMs, produces an iteration plan for enterprise integration, and provides cost estimates. Through Chapter V's results it can be concluded that C-DSMs are a viable method to producing integration plans and reducing complexity through integration. Furthermore, it can be seen that traditional DSM methods can be re-utilized in yet new directions to provide more utility to practitioners as well as the academic community.

The primary objective of this research was to develop an algorithm-based method that can optimally integrate a large number of enterprise IT systems over time. Through the course of accomplishing this objective, supporting theories and methods were discovered, which led to the creation of an algorithm that can provide a user-customizable, stepwise path to systems integration.

In developing the C-DSM method it was found that other methods are available such as direct implementation of an ERP software solution or enterprise transformation efforts

such as BPR, but none of them combined the needs presented in this research such as algorithmic, repeatable, and with customizable speed towards information systems integration. It was also found that there are many factors that can define the relationship between systems and there are many costs that can be associated with integration, but through customizable parameters, that it is not necessary to develop all of these parameters into the model when users can enter and define them as necessary. Additionally, it was found that the speed and format of integration is perceivably different from the private sector and the government sector. Private sector integration is free to integrate faster and more directly with solutions purchased "off the shelf" whereas the government is severely hindered with enormous problem spaces, conflicting direction from politics and laws, and the resultant need to integrate slowly as all of the factors must be addressed through painstakingly review of all of the factors prior to any attempt at integration. Finally, pros and cons of complete integration versus spiral development integration were not fully explored as it was found that current government efforts will invariably take the spiral approach as the factors previously mentioned hold great impact on decision making and implementation.

Significance of Research

The implications of this research come as a direct answer to AF problems and shortfalls as identified in Chapters I and II, as well as provide a new integration method to other government and private sector enterprises. The C-DSM method described in this dissertation is a new method using static DSMs that collapse through integration and iteration. Through clustering within user-defined constraints, the speed and cost of integration can be controlled and studied. This research also combined successful advanced

decision analysis techniques to develop weighted DSMs to produce a useful "AS-IS" state of the enterprise. As determined through an extensive literature review, this research is possibly the first to use traditional DSMs in a longitudinal manner to reduce complexity, and has at least contributed to the dearth of DSM use for enterprise problems. Legacy IS were modeled in the examples in this dissertation using C-DSMs, and are different than product describing DSMs. C-DSMs through exploration of logical connections, extends product-centered DSM application and show promise in the realm of enterprise solutions.

The extension of DSMs to enterprise level problems contributes to the theoretical literature of general systems theory, networking theory, and complexity theory. The problem of individual information system creation within disconnected siloes provided in Chapters 1 and 2 contributes to general systems theory as they describe systems (silo managers) that had to react to their environment (competition). Networking theory is also supported in this research as DSM entries represent logical connections and are then reduced through integration and iteration. Complexity theory is addressed as the legacy information systems are subsystems of a much larger SoS and react to their environment, but will evolve towards a common goal of the SoS.

This dissertation research fills in the USAF integration gaps identified in Chapter II and allows for an enterprise-focused evaluation of systems that produces an "AS-IS" state of the architecture. The method also provides a user-defined set of constraints to control the speed of integration allowing users to model perceived challenges in the integration process. The method primary achievement, however, is the development of collapsing matrices that, through iteration, produce integration plans. These integration plans provide a roadmap for

system architects, integration offices, and CEOs to see and plan for achievement of a desired "TO-BE" state of the enterprise.

Recommendations for Action

The AF should recycle some of the data they have acquired from previous integration efforts and determine whether or not the C-DSM method would have produced a more direct and systematic plan. A common practice in the corporate sector is to select a single ERP and employ a cutover strategy (many-to-one transition strategy), but many ERP implementations fail or minimally succeed (Umble, 2003) lending support to the idea that there should be exploration of other alternatives like the C-DSM methodology. The C-DSM method allows for any range of integration set by the user (many-to-many), but has been tested in this research from 10 – 50% reduction per iteration. Slower alternatives may be necessary when the number of systems is large and this slower alternative allows for interim solutions. As government evolution is stymied by politics, policies, laws, culture, complex hierarchies, etc., long term planning seems more appropriate. Additionally, long term algorithmically produced transition plans may mitigate some challenges presented by frequent leadership changes that are common in the USAF. The USAF should incorporate this method as an alternative plan generator to any current transition plans to assist in any integration efforts where system redundancy is expected, and a future state of few, or only one, system is desirable.

Recommendations for Future Research

As this research introduces an area that has, as of yet, been insufficiently explored, more research in peer reviewed literature needs to be conducted. Research exploring DSMs in the realm of enterprise-focused research could certainly expand to logistics, supply chain management, and other disciplines that focus on "the big picture." A new method of collapsing DSMs was introduced in this paper and most assuredly needs to be further examined. There should be research conducted to see if there are differences in controlled vs. uncontrolled enterprise governance structures. In the government sector, there is ultimately a level of control that encompasses the entire enterprise, but private sector enterprises may contain a conglomerate of separately owned and operated corporations. Determining whether or not collapsing DSMs can model specific systems that should never integrate even at the most extreme conditions should be investigated. More robust economic techniques are needed to increase the applicability of this method such as NPV and the inclusion of other types of costs incurred by enterprises in the midst of integration. In Chapter IV the C-DSM method clustered on integration cost and in Chapter V it expanded to a weighted sum of factors reflecting the relationship strength between two systems, but expanding the clustering decisions to multiple factors/matrices would present a more robust technique.

As the C-DSM method aims at reducing the number of elements in a system based on the notion of redundancy, perhaps it can also apply to manpower reductions and the identification of cross-training opportunities for organizations desiring to reduce their human capital investment.

A key feature of the C-DSM method was the use of multipliers to aid users in estimating future systems in their enterprise. If the relationships of current and planned states are not understood then perhaps randomizing the multipliers when data is not available to support a decision could provide insightful results.

The C-DSM method was developed using MATLAB and simulated annealing, but perhaps other search optimization methods can be applied to cluster and re-sort DSMs. One such search method would be the multi-objective knapsack problem where, given a set amount of resources (money), and the set of constraints discussed in Chapter V to include cluster size (a form of bins –aka knapsacks), the amount of integration could be "purchased" during each iteration.

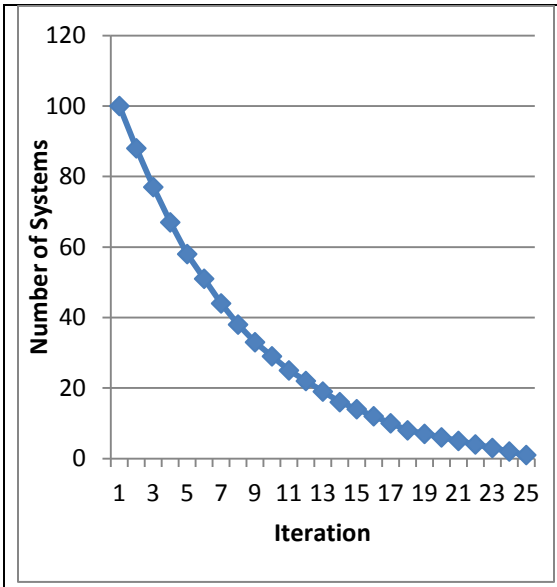
Finally, there should be comprehensive case studies from both the government and private sectors that include more than cost estimates to determine the validity and utility of this model at the application level.

Summary

In summary, this research described an AF specific problem that is shared with the private sector in the form of enterprise integration challenges when faced with a large complex network of redundant systems in Chapter I. The problem was defined and framed in a body of theories that formed the basis for this research in Chapter II. Once the theoretical basis was developed, available tools from those theories were explored until one specific tool was chosen to mitigate the defined problem in Chapter III. However, the tool in its current state was incapable of reducing complexity or redundancy in a long term, iterative manner

that met all of the needs of the AF. Therefore, the C-DSM method was developed and explained in detail in Chapters IV and V revealing an answer to the AF's integration failures.

Appendix A. Complete Baseline Integration Plan Example

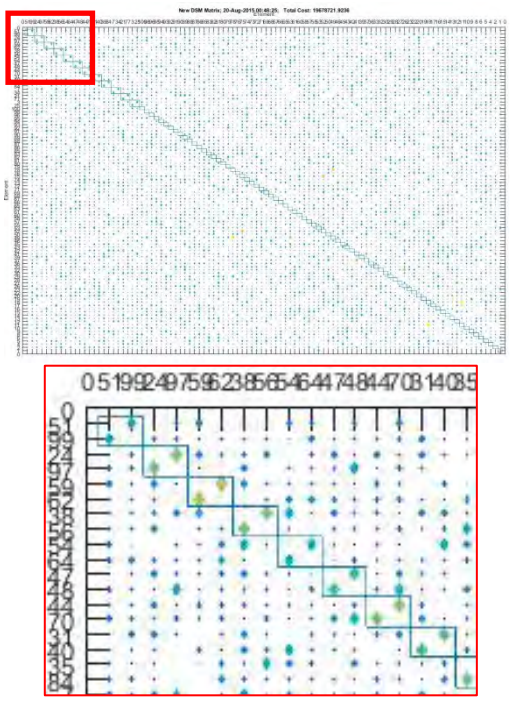


*Element identifying labels are recycled each iteration meaning system 2 during iteration 1 is not necessarily the same as system 2 during iteration 2. This shortcoming will be corrected in a future upgrade to the algorithm.

Iteration	Systems Clustered*
1	79/92; 32/89; 72/82; 56/80; 22/76; 59/91; 10/43; 7-38; 26/46; 21/77; 18/45; 16/60;
2	70/86; 27/84; 20/81; 48/65; 47/63; 26/62; 34/55; 3/51; 39/64; 12/24; 11/80;
3	43/63; 20/62; 88/46; 36/37; 18/35; 28/75; 23/33; 19/50; 17/42;
4	27/30/41; 39/47; 33/35; 7/32; 20/42; 18/55; 16/63;
5	3/21/41; 2/44; 13/43; 4/38; 9/29; 22/40;
6	32/37; 3/34; 28/31; 2/25; 12/22; 15/18; 14/48;;
7	2/38; 24/39; 23/34; 17/25; 13/41; 10/36;
8	19/32; 13/30; 8/14; 4/34; 1/18;
9	16/29; 21/26; 6/28; 1/11;
10	18/21; 16/19; 15/25; 13/14;
11	18/21; 5/14; 8/22;
12	14/19; 4/16; 5/9;
13	7/18; 6/9; 2/11;
14	11/15; 10/12;
15	6/10; 3/5;
16	3/8; 1/9;
17	5/8; 4/9;
18	3/7;
19	4/7;
20	4/5;
21	1/4;
22	2/3;
23	1/3;
24	1/2;

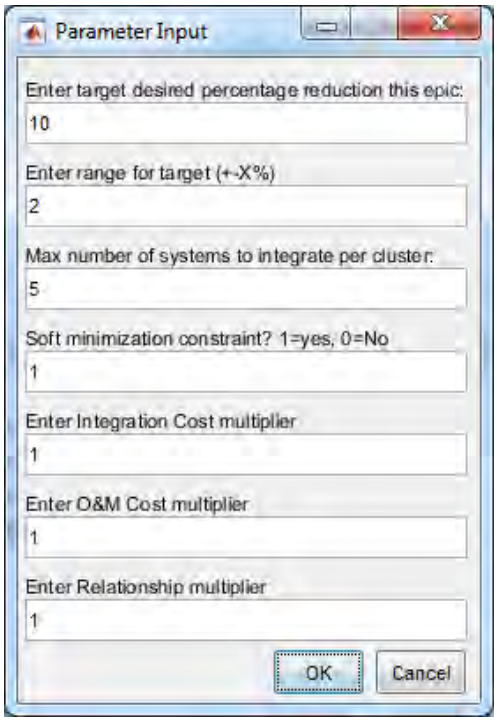
Iteration	Number of Systems	Percent Reduction Achieved	Integration Cost this Iteration (in $millions)	O&M Costs Removed this Iteration (in $millions)
1	88	12%	439.2	53.5
2	77	13%	393.7	50.8
3	67	13%	261.3	47.1
4	58	13%	336.3	50.5
5	51	12%	313.4	28.5
6	44	14%	213.3	38.0
7	38	14%	195.1	29.5
8	33	13%	169.4	28.0
9	29	12%	141.4	21.6
10	25	14%	160.2	17.8
11	22	12%	99.5	16.7
12	19	14%	113.9	10.6
13	16	16%	89.2	12.1
14	14	13%	64.8	10.4
15	12	14%	65.0	8.4
16	10	17%	66.4	8.5
17	8	20%	64.0	10.2
18	7	13%	31.6	3.8
19	6	14%	29.5	5.5
20	5	17%	32.8	4.5
21	4	20%	34.6	4.8
22	3	25%	28.2	5.2
23	2	33%	33.1	5.0
24	1	50%	33.0	4.4

Cumulative Integration Cost of Solution: $3.4B Total Cost after 25 years (cumulative integration plus cumulative diminishing O&M): $6.9B	O&M Costs Baseline (do nothing 25 years): $11.98 B ($479M annual) New O&M Costs: $3.6M annually Total Savings of Solution ($11.98B – $6.9B): $5.08B



Iteration 1

In the first iteration, it can be seen that the larger diamonds, which represent a stronger relationship, are pulled into clusteres with other systems where they are strongly connected or functionaly similar based on the user inputs of the relationship DSM.



Reducing Enterprise Systems Complexity Through Integration Using Collapsing Design Structure Matrices

Capt Michael Kretser
Advisor: Dr. Jeffrey Ogden
Committee: Dr. John Colombi, Dr. Paul Hartman
Department of Operational Sciences (ENS)
Air Force Institute of Technology

INTRODUCTION

Since the Industrial Revolution product creation has evolved from master craftsmen to segmented processes of mass production leading to departments of specialization and the growth of hierarchy based business models.

The introduction of non-networked computer technology led to optimization of *departmental performance*.

The Information Age brought forth opportunities for *enterprise performance* through networking technology.

As companies merge and/or expand legacy systems built in isolation are rarely merged or integrated and remain in use while performing similar functionality for separate departments.

It is not uncommon for enterprises to have minimal visibility and understanding of enterprise systems' relationships resulting in a web of hidden costs in duplication of effort.

RESEARCH GOALS

The goal of this research was to develop a mathematical algorithm that produced a roadmap for the reduction of complexity of an enterprise network using a novel application of DSM concepts.

This method displays the "As-is" state of enterprise systems and suggests a stepwise progression to obtain a logical "To-be" architecture for the purpose of strategic enterprise decision making.

Original Thebeau Algorithm

$$\text{Total Coordination Cost} = \sum_{k=1..z} \sum_{i,j=1..n} (ExtraClusterCost_{ijk} + IntraclusterCost_{ijk})$$

where:

$$ExtraClusterCost_{ijk} = \begin{cases} (x_{ij} + x_{ji}) * n^{power} & \text{if } c_{ik} \neq c_{jk} \\ 0, & \text{otherwise} \end{cases}$$

$$IntraClusterCost_{ijk} = \begin{cases} (x_{ij} + x_{ji}) * ClusterSize_k{}^{power} & \text{if } c_{ik} = c_{jk} \\ 0, & \text{otherwise} \end{cases}$$

z = number of clusters in the solution

n = number of elements (number of systems in the matrix)

Matrix $\mathbf{C} = \{c_{ik}\}, i = 1..n, k = 1..p$ determines which system i is assigned to each cluster k

$$c_{ik} = \begin{cases} 1, & \text{if system } i \text{ is in the cluster } k \\ 0, & \text{otherwise} \end{cases}$$

C-DSM

Constraints	
$\sum_{i=1}^{m} c_{ik} \leq \tau$	$\sum_{k=1}^{p} c_{ik} \geq \delta$
$\sum_{k=1}^{p} c_{ik} \leq \varepsilon$	$\sum_{i=1}^{p} c_{ij} = 1$

$$\text{Collapsed Matrix Entry} = \frac{\sum_{ij=1}^{k} x_{ij} + x_{ji}}{k} * \lambda$$

τ = *user defined maximum cluster size*
δ = *user defined minimum number of clusters*
ε = *user defined maximum number of clusters*
λ = *user defined multiplier after integration, (<1 indicates weaker relationship after integration, <1 indicates weaker relationship)*

Reduce Complexity through Integration Example C-DSM

	SysA	SysB	SysC	SysD	SysE	SysF	SysG	SysH	SysI	SysJ
SysA	1	0.5	1	0	0.25	0.75	0	1	1	1
SysB	0.5	1	0	1	1	0.25	0.75	0	0	0.5
SysC	1	0	1	0.5	0.25	0.25	0.25	0.75	1	1
SysD	0	1	0.5	1	0	0	0	1	0.25	0.5
SysE	0.25	1	0.25	0	1	0.5	0.25	1	1	0
SysF	0.75	0.25	0.25	0	0.5	1	0.1	0.3	0.6	1
SysG	0	0.75	0.25	0	0.25	0.1	1	0	0	0
SysH	1	0	0.75	1	1	0.3	0	1	0	1
SysI	1	0	1	0.25	1	0.6	0	0	1	0.7
SysJ	1	0.5	1	0.5	0	1	0	1	0.7	1

→

	Cls1	Cls2	Cls3	Cls4	Cls5	Cls6	Cls7	Cls8
Cls1	0.5	0.5	0.6	0.5	0.4	0.5	0.5	0.8
Cls2	0.5	0.5	0.1	0.1	0.1	0.9	0.6	0.8
Cls3	0.6	0.1	0	0.5	0.3	1	1	0
Cls4	0.5	0.1	0.5	0	0.1	0.3	0.6	1
Cls5	0.4	0.1	0.3	0.1	0	0	0	0
Cls6	0.5	0.9	1	0.3	0	0	0	1
Cls7	0.5	0.6	1	0.6	0	0	0	0.7
Cls8	0.8	0.8	0	1	0	1	0.7	0

Used to calculate next iteration's estimated entries
" and used to calculate ExtraClusterCost
" and used to calculate ExtraClusterCost
" and used to calculate IntraClusterCost, Integration Cost

*using averages, no modifiers or penalties

CONTRIBUTIONS

- Introduced the first "Collapsing DSMs"
- Static DSMs are populated using weighted value tables, cluster through user-defined constraints, and collapse through integration and iteration
- Developed an unbiased analytical plan for the integration and/or cancellation of redundant systems
- Developed an automated "To-be" architecture based on input factors, desired number of end state systems, and the current "As-is" architecture

FUTURE RESEARCH

- Add more robust economic cost calculations
- Expand the number of clustering factors
- Conduct a case study applying this method to Air Force logistics enterprise systems data
- **Apply the C-DSM method to supply chains to determine the validity of the method in an uncontrolled enterprise**
- **Study the applicability to manpower reduction /cross-training**

Sponsor
AFMC/A4N Systems Integration Office of the Logistics and Sustainment Directorate

DEPARTMENT OF OPERATIONAL SCIENCES

Bibliography

Ahmad, M. M., & Cuenca, R. P. (2013). Critical success factors for ERP implementation in SMEs. *Robotics and Computer-Integrated Manufacturing, 29*(3), 104-111.

Alexanderson, G. L. (2006). About the cover: Euler and Konigsberg's bridges: A historical view. *Bulletin of the American Mathematical Society, 43*(4), 567-573.

Al-Fawaz, K., Al-Salti, Z., & Eldabi, T. (2008). Critical success factors in ERP implementation: A review. *Proceedings of the European and Mediterranean Conference on Information Systems,* Dubai: EMCIS.

Aoyama, M., & Tanabe, H. (2011). A design methodology for real-time distributed software architecture based on the behavioral properties and its application to advanced automotive software. Paper presented at the *Software Engineering Conference (APSEC), 2011 18th Asia Pacific,* Ho Chi Minh. 211-218.

Baharum, Z., Ngadiman, M. S., & Haron, H. (2009). *Critical factors to ensure the successful of OS-ERP implementation based on technical requirement point of view* Retrieved from http://gateway.webofknowledge.com/gateway/Gateway.cgi?&GWVersion=2&SrcAuth=SerialsSolutions&SrcApp=360&DestLinkType=FullRecord&DestApp=INSPEC&KeyUT=INSPEC:10717281

Bartolomei, J. E., Hastings, D. E., de Neufville, R., & Rhodes, D. H. (2012). Engineering systems multiple-domain matrix: An organizing framework for modeling large-scale complex systems. *Systems Engineering, 15*(1), 41-61.

Batallas, D. A., & Yassine, A. A. (2006). Information leaders in product development organizational networks: Social network analysis of the design structure matrix. *IEEE Transactions on Engineering Management, 53*(4), 570.

Baxter, G. (2010). *Key issues in ERP system implementation*. University of St. Andrews: University of St. Andrews, School of Computer Science.

Bezuidenhout, C. N. (2012). Network-analysis approaches to deal with causal complexity in a supply network. *International Journal of Production Research, 50*(7), 1840.

Bidgoli, H. (2004). The internet encyclopedia. (pp. 707) John Wiley& Sons, Inc.

Bilalis, N., & Maravelakis, E. (2006). Analysing the dependencies between the innovation attributes in NPD using design structure matrix - the I-DSM tool. *International Journal of Product Development, 3*(3-4), 432-446.

Bliss, G. R. (2013). *Root cause analysis of the expeditionary combat support system program*. Washington DC: Performance Assessments and Root Cause Analyses (PARCA).

Brown, N., Nord, R. L., Ozkaya, I., & Pais, M. (2011). Analysis and management of architectural dependencies in iterative release planning. Paper presented at the *Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on,* Boulder, CO. 103-12.

Browning, T. R. (2001a). Applying the design structure matrix to system decomposition and integration problems: A review and new directions. *IEEE Transactions on Engineering Management, 48*(3), 292-306.

Cai, Y., Iannuzzi, D., & Wong, S. (2011). Leveraging design structure matrices in software design education. *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T),* 179-88.

Carriere, J., Kazman, R., & Ozkaya, I. (2010). A cost-benefit framework for making architectural decisions in a business context. Paper presented at the *Software Engineering, 2010 ACM/IEEE 32nd International Conference on,* Cape Town. *, 2* 149-157.

Chang, C., Chiang, D. M., Wu, A. W., & Chang, Y. (2011). Identifying critical features of complex systems under prior known conditions. *Computers & Industrial Engineering, 61*(3), 788-793.

Chen, C. C., Law, C. C., & Yang, S. C. (2009). Managing implementation failure: A project management perspective. *IEEE Transactions on Engineering Management, 56*(1), 159-170.

Chen, Y., Cheng, P., & Yin, J. (2010). Change propagation analysis of trustworthy requirements based on dependency relations. Paper presented at the *Information Management and Engineering (ICIME), 2010 the 2nd IEEE International Conference on,* Chengdu. 246-251.

Chief Information Officer Council. (2001). *A practical guide to federal enterprise architecture v 1.0.* Government Accountability Office

Choi, T. Y., Wu, Z., Ellram, L., & Koka, B. R. (2002). Supplier-supplier relationships and their implications for buyer-supplier relationships. *Engineering Management, IEEE Transactions on, 49*(2), 119-130.

Collins, J. R., & Krause, B. (2005). *Theater battle management core system: Systems engineering case study.* (CSE Report). Air Force Institute of Technology: Center for Systems Engineering.

Dagli, C. H., Ergin, N., Enke, D., Giammarco, K., Gosavi, A., Qin, R., . . . Colombi, J. (2013). *An advanced computational approach to system of systems analysis & architecting using agent-based behavioral model.* (Technical Report No. SERC-2013-TR-021-3).Systems Engineering Research Center. (TR-021)

Deng, X., Huet, G., Tan, S., & Fortin, C. (2012). Product decomposition using design structure matrix for intellectual property protection in supply chain outsourcing. *Computers in Industry, 63*(6), 632-641.

Dianting, L., Liu Y., & Lei, X. (2013). Networked collaborative design task decomposition based on LDSM. Paper presented at the *Conference Anthology, IEEE,* China. 1-4.

Engel, A., & Browning, T. R. (2008). Designing systems for adaptability by means of architecture options. *Systems Engineering, 11*(2), 125-146.

Eppinger, S. D., & Browning, T. R. (2012). *Design structure matrix methods and applications*. Cambridge, MA: The MIT Press.

Farid, A. M., & McFarlane, D. C. (2006). An approach to the application of the design structure matrix for assessing reconfigurability of distributed manufacturing systems. Paper presented at the *Distributed Intelligent Systems: Collective Intelligence and its Applications, 2006. DIS 2006. IEEE Workshop on,* Prague. 121-126.

Feng, Q., Zhu, C., Sun, B., & Liu, L. (2011). Process reengineering method for synthesis design of reliability maintainability supportability and performance. Paper presented at the *Reliability, Maintainability and Safety (ICRMS), 2011 9th International Conference on,* Guiyang. 826-832.

Fernandez, C. I. G. (1998). *Integration analysis of product architecture to support effective team co-location* (MS).

Ford, T. C., Colombi, J. M., Jacques, D. R., & Graham, S. R. (2009a). A general method of measuring interoperability and describing its impact on operational effectiveness. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, 6*(1), 17-32. Retrieved from

Ford, T. C., Colombi, J. M., Jacques, D. R., & Graham, S. R. (2009b). On the application of classification concepts to systems engineering design and evaluation. *Systems Engineering, 12*(2), 141-154. doi:10.1002/sys.20114

Fu X., Song M., Yu Y., & Mian, C. (2010). Integration management view of information systems in an enterprise. Paper presented at the *Information Technology and Computer Science (ITCS), 2010 Second International Conference on,* Kiev. 317-321.

Fu, Y., Li, M., & Chen, F. (2012). Impact propagation and risk assessment of requirement changes for software development projects based on design structure matrix. *Int. J. Proj. Mgmt,* (30), 363-373.

Gerstner, L. V. J. (2002). *Who says elephants can't dance?* New York City: HarperCollins.

Grebe, M., & Danke, E. (2013). *Simplify IT: Six ways to reduce complexity* The Boston Consulting Group.

Guenov, M. D., & Barker, S. G. (2005). Application of axiomatic design and design structure matrix to the decomposition of engineering systems. *Systems Engineering, 8*(1), 29-40.

Hameri, A. P., Nihtila, J., & Rehn, J. (1999). Document viewpoint on one-of-a-kind delivery process. *International Journal of Production Research, 37*(6), 1319-1336.

Hammer, M., & Champy, J. (2003). *Reengineering the corporation*. New York City: HarperCollins.

Hamraz, B., Caldwell, N. H. M., & Clarkson, P. J. (2013a). A matrix-calculation-based algorithm for numerical change propagation analysis. *Engineering Management, IEEE Transactions on, 60*(1), 186-198.

Hamraz, B., Caldwell, N. H. M., & Clarkson, P. J. (2013b). A holistic categorization framework for literature on engineering change management. *Systems Engineering, 16*(4), 473-505.

Helmer, R., Yassine, A., & Meier, C. (2010). Systematic module and interface definition using component design structure matrix. *Journal of Engineering Design, 21*(6), 647-675.

Holley, V., Jankovic, M., & Yannou, B. (2014). Physical interface ontology for management of conflicts and risks in complex systems. *Concurrent Engineering-Research and Applications, 22*(2), 148-161.

Honour, E. C., & Browning, T. R. (2007). Dynamic optimization of systems of systems using value measurement. *Journal of Integrated Design & Process Science, 11*(2), 33-53.

Inspector General of the United States Department of Defense. (2012). *Enterprise resource planning systems schedule delays and reengineering weaknesses increase risks to DoD's auditability goals.* (DODIG Report No. DODIG-2012-111).

Jankovic, M., Holley, V., & Yannou, B. (2012). Multiple-domain design scorecards: A method for architecture generation and evaluation through interface characterisation. *Journal of Engineering Design, 23*(10-11), 743-763.

Jin-long C., Yi-yong X., & Xiao-yan X., (2011). A life cycle cost-based product design approach through integral architecture. Paper presented at the *Industrial Engineering and Engineering Management (IE&EM), 2011 IEEE 18Th International Conference on,* Changchun. 526-531.

Johnson, A. C., Ogden, J. A., & and Ford, T. C. (2010). A method of measuring supply chain interoperability. *Proceedings of the 2010 Institute of Industrial Engineering Research Conference,* Cancun, Mexico.

Kamrani, A. K., & Azimi, M. (2011). *Systems engineering tools and methods* (1st ed.) CRC Press, Boca Raton, FL.

Kasperek, D., Fink, S., Maisenbacher, S., Bauer, W., & Maurer, M. (2014). Assessing the informative value of complexity metrics within design structure matrices in early development phases of complex systems. Paper presented at the 351-356.

Kauffman, S. (1995). *At home in the universe: The search for the laws of self-organization and complexity*. New York; Oxford: Oxford University Press.

Keeney, R. (1992). *Value-focused thinking: A path to creative decision making*. Cambridge, MA: Harvard University Press.

Kirkwood, C. (1992). *Strategic decision making: Multiobjective decision analysis with spreadsheets*. Belmont, CA: Wadsworth Publishing Company.

Koch, J., Maisenbacher, S., Maurer, M., Reinhart, G., & Zäh, M. F. (2014). Structural modeling of extended manufacturing systems – an approach to support changeability by reconfiguration planning. *Procedia CIRP, 17*(0), 142-147.

Kuqi, K., Eveleigh, T., Holzer, T., & Sarkani, S. (2012). Using design structure matrix for improving electronic medical record usability. Paper presented at the *Systems Conference (SysCon), 2012 IEEE International,* Vancouver, BC. 1-6.

Lagerstrom, R., Baldwin, C., MacCormack, A., & Aier, S. (2013). *Visualizing and measuring enterprise application architecture: An exploratory telecom case.* (No. 13-103).Harvard Business School Working Paper.

Lagerstrom, R., Baldwin, C., MacCormack, A., & Aier, S. (2014). *Visualizing and measuring enterprise application architecture: An exploratory telecom case*

Lambe, A., & Martins, J. (2012). Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural & Multidisciplinary Optimization, 46*(2), 273-284.

Lambert, D. M., & Cooper, M. C. (2000). Issues in supply chain management. *Industrial Marketing Management, 29*(1), 65-83.

Lancaster, J., & Cheng, K. (2008). A fitness differential adaptive parameter controlled evolutionary algorithm with application to the design structure matrix. *International Journal of Production Research, 46*(18), 5043-5057.

Lee, H., Padmanabhan, V., & Whang, S. (1997). Information distortion in supply chains: The bullwhip effect. *Manage. Sci., 43*, 546-558.

Lee, H. C. (July, 2009). Network theory introduction: Presentation given at the 13th Taiwan nuclear physics summer school national chiao-tung university. Retrieved from http://sansan.phy.ncu.edu.tw/~hclee/lec/NetworkTheory-V3.pdf

Lefever, G., Pesanello, M., Fraser, H., & Taurman, L. (2011). *Life science: Fade or flourish?* IBM Institute for Business Value.

Li, S., & Chen, L. (2014). Identification of clusters and interfaces for supporting the implementation of change requests. *IEEE Transactions on Engineering Management, 61*(2), 323-335.

Liang, L. Y. (2009). Grouping decomposition under constraints for design/build life cycle in project delivery system. *International Journal of Technology Management, 48*(2), 168-187.

Lopes, C. V., Bajracharya, S. K., Rashid, A., & Aksit, M. (2006). Assessing aspect modularizations using design structure matrix and net option value. *Transactions on Aspect-Oriented Software Development I, 3880*, 1-35.

Luna, S., Lopes, A., Tao, H. Y. S., Zapata, F., & Pineda, R. (2013). Integration, verification, validation, test, and evaluation (IVVT&E) framework for system of systems (SoS). *Procedia Computer Science, 20*(0), 298-305.

Madni, A. M., & Moini, A. (2007). Viewing enterprises as systems-of-systems (SoS): Implications for SoS research. *Journal of Integrated Design & Process Science, 11*(2), 3.

Maier, M. W. (1998).
Architecting principles for systems-of-systems *Systems Engineering, 1*(4), 267-284.

Malmstrom, J., Pikosz, P., & Malmqvist, J. (1999). Complementary roles of IDEF0 and DSM for the modeling of information management processes. *Concurrent Engineering-Research and Applications, 7*(2), 95-103.

Manson, S. M. (2001). Simplifying complexity: A review of complexity theory. *Geoforum, 32*(3), 405-414.

McNerney, J., Farmer, J. D., Redner, S., & Trancik, J. E. (2011). Role of design complexity in technology improvement. *Proceedings of the National Academy of Sciences of the United States of America, 108*(22), 9008-9013.

Mikaelian, T., Rhodes, D. H., Nightingale, D. J., & Hastings, D. E. (2012). A logical approach to real options identification with application to UAV systems. *IEEE Trans Syst Man Cybern A Syst Hum, 42*(1), 32-47.

Motwani, J., Subramanian, R., & Gopalakrishna, P. (2005). Critical factors for successful ERP implementation: Exploratory findings from four case studies. *Computers in Industry, 56*(6), 529-544.

Nakata, Y. (2009). Business architecture and dynamics of interdependences analyzed by design structure matrix: Comparing liquid crystal display and semiconductor industries. Paper presented at the *Portland International Conference on Management of Engineering & Technology, 2009. PICMET 2009.* Portland, OR. 3286-3295.

Pathak, S. D. (2007). Complexity and adaptivity in supply networks: Building supply network theory using a complex adaptive systems perspective. *Decision Sciences, 38*(4), 547-580.

Pimmler, T. U., & Eppinger, S. D. (1994). Integration analysis of product decomposition. *ASME Design Theory and Methodology Conference,* Minneapolis, MN, USA.

Pineda, R. L., & Smith, E. D. (2011). Functional analysis and architecture. In A. K. Kamrani, & M. Azimi (Eds.), *Systems engineering tools and methods* (1st ed., pp. 35-79) CRC Press, Boca Raton, FL.

Ponomarov, S. Y., & Holcomb, M. C. (2009). Understanding the concept of supply chain resilience. *International Journal of Logistics Management, 20*(1), 124-143.

Rebovich Jr., G. (2008). Enterprise system of systems. In M. Jamshidi (Ed.), *Systems of systems engineering: Principles and applications* (pp. 165-189). Boca Raton, FL: CRC Press.

Reilly, S. (2012) How the air force blew $1B on a dud system. Retrieved from http://www.federaltimes.com/article/20121126/DEPARTMENTS01/311260009/How-Air-Force-blew-1-billion-dud-system

Riposo, J., Weichenberg, G., Duran, C. K., & Fox, B. (2013). *Improving air force enterprise resource planning-enabled business transformation.* (RAND Report No. RAND_RR250).

Roghe, F., Toma, A., Messenbock, R., Kempf, S., & Marchingo, M. (2013, Nov.). Breaking free of the silo: Creating lasting competitive advantage through shared services. *BCG Perspectives,*

Sanders, N. R., Autry, C. W., & Gligor, D. M. (2011). The impact of buyer firm information connectivity enablers on supplier firm performance : A relational view. *International Journal of Logistics Management, 22*(2), 179-201.

Sharman, D. M., & Yassine, A. A. (2007). Architectural valuation using the design structure matrix and real options theory. *Concurrent Engineering-Research and Applications, 15*(2), 157-173.

Shi-Jie Chen, & Huang, E. (2007). A systematic approach for supply chain improvement using design structure matrix. *Journal of Intelligent Manufacturing, 18*(2), 285-299.

Shoviak, M. J. (2001). *Decision analysis methodology to evaluate integrated solid waste management alternatives for a remote Alaskan air station* (Master's Thesis).

Simpson, J. J., & Simpson, M. J. (2009). System of systems complexity identification and control. Paper presented at the *IEEE International Conference on System of Systems Engineering.* Albuquerque, NM. 1-6.

Simpson, J., & Simpson, M. (2011). Complexity reduction: A pragmatic approach. *Systems Engineering, 14*(2), 180-192.

Sosa, M. E., Eppinger, S., & Rowles, C., (2003). Identifying modular and integrative systems and their impact on design team interactions. *Journal of Mechanical Design, 125*(2), 240-252.

Spaulding, C. R., Gibson, W. S., Schreurs, S. F., Linsenbardt, D. L., & DeSimone, A. (2011). Systems engineering for complex information systems in a federated, rapid development environment. *Johns Hopkins APL Technical Digest, 29*(4), 310-326.

Steward, D. V. (1981). The design structure system: A method for managing the design of complex systems . *IEEE Transactions on Engineering Management, 28*(3), 71-74.

Strachan, E. V. (2008). *Governance structure transformation during ERP implementations* (Master's Thesis)

Su-Yu Liu, Li, C., Yi-Ping Feng, & Rong, G. (2013). Clustering structure and logistics: A new framework for supply network analysis. *Chemical Engineering Research & Design: Transactions of the Institution of Chemical Engineers Part A, 91*(8), 1383.

Tang, D., Zhang, G., & Dai, S. (2009). Design as integration of axiomatic design and design structure matrix. *Robotics and Computer-Integrated Manufacturing, 25*(3), 610-619.

Tang, D., Zhu, R., Dai, S., & Zhang, G. (2009). Enhancing axiomatic design with design structure matrix. *Concurrent Engineering: Research and Applications, 17*(2), 129-137.

Thebeau, R. E. (2001). *Knowledge management of system interfaces and interactions from product development processes* (MS).

U.S. Government Accountability Office. (2007). *Homeland security: Departmentwide integrated financial management systems remain a challenge.* (GAO Report No. GAO-07-536).

U.S. Government Accountability Office. (2008). *DoD business transformation: Air Force's current approach increases risk that asset visibility goals and transformation priorities will not be achieved: Report to the subcommittee on readiness and management support, committee on armed services, U.S. senate.* (GAO Report No. GAO-08-866).

U.S. Government Accountability Office. (2013). *Department of homeland security: Progress made and work remaining after nearly 10 years in operation.* (GAO Report No. GAO-13-370T).

Umble, E. J. (2003). Enterprise resource planning: Implementation procedures and critical success factors. *European Journal of Operational Research, 146*(2), 241-257.

van Beek, T. J., Erden, M. S., & Tomiyama, T. (2010). Modular design of mechatronic systems with function modeling. *Mechatronics, 20*(8), 850-863.

Von Bertalanffy, L. (1951). Theoretical models in biology and psychology. *Journal of Personality, 20*(1), 24.

Von Bertalanffy, L. (1972). The history and status of general systems theory. *Academy of Management Journal, 15*(4), 407-426.

Von Bertalanffy, L. (2008). An outline of general system theory. *Emergence: Complexity & Organization, 10*(2), 103-116.

Waldrop, M. (1992). *Complexity: The emerging science at the edge of order and chaos*. Touchstone, New:

Wang, E. T. G., Shih, S., Jiang, J. J., & Klein, G. (2008). The consistency among facilitating factors and ERP implementation success: A holistic view of fit. *Journal of Systems and Software, 81*(9), 1609-1621.

Wehrwein, B. (2013). Lightweight software reverse engineering using augmented matrix visualizations. Paper presented at the *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on,* Eindhoven. 1-4.

Wen-Tin Lee, Kuo-Hsun Hsu, & Lee, J. (2012). Designing software architecture with use case blocks using the design structure matrix. Paper presented at the *2012 International Symposium on Computer Consumer and Control (IS3C),* Taichung. 654-7.

Wen-Tin Lee, Whan-Yo Deng, Lee, J., & Shin-Jie Lee. (2010). Change impact analysis with a goal-driven traceability-based approach. *International Journal of Intelligent Systems, 25*(8), 878-908.

Xia, W., & Lee, G. (2005). Complexity of information systems development projects: Conceptualization and measurement development. *Journal of Mgt Info Sys, 22*(1), 45-83.

Xiaogang, X., Chao, L., Jian, Y., & Yahua, C. (2006). An analytical method based on design structure matrix for modular identification. Paper presented at the *7th International Conference on Computer-Aided Industrial Design and Conceptual Design, 2006. CAIDCD '06,*Hangzhou. 1-4.

Yee, S. L. (2007). Differing roles of axiomatic design and design structure matrix in reducing system complexity. Paper presented at the *IEEE International Conference on Industrial Engineering and Engineering Management, 2007,* Singapore. 994-8.

Yong-hui Guo. (2010). Modeling for design chain optimization based on information flow. Paper presented at the *Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference on,* Xiamen. 1551-1554.

Zachman, J. A. (2010). A framework for information systems architecture. *IBM Systems Journal, 26*(3), 276-292. doi:10.1147/sj.263.0276

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704–0188

1. REPORT DATE (DD–MM–YYYY)	2. REPORT TYPE	3. DATES COVERED (From — To)
17-09-2015	Dissertation	Aug 2012 – Sep 2015

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Building Enterprise Transition Plans Through The Development Of Collapsing Design Structure Matrices	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Kretser, Michael P., Captain, USAF	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENS-DS-15-S-033
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ Air Force Material Command Systems Integration Division 4375 Chidlaw Road WPAFB OH 45433 POC: Intentionally left blank	10. SPONSOR/MONITOR'S ACRONYM(S) HQ AFMC/A4N
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A. Approved For Public Release; Distribution Unlimited.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

The United States Air Force (USAF), like many other large enterprises, has evolved over time, expanded its capabilities and has developed focused, yet often redundant, operational silos, functions and information systems (IS). Recent failures in enterprise integration efforts herald a need for a new method that can account for the challenges presented by decades of increases in enterprise complexity, redundancy and Operations and Maintenance (O&M) costs. Product or system-level research has dominated the study of traditional Design Structure Matrices (DSMs) with minimal coverage on enterprise-level issues. This research proposes a new method of collapsing DSMs (C-DSMs) to illustrate and mitigate the problem of enterprise IS redundancy while developing a systems integration plan. Through the use of iterative user constraints and controls, the C-DSM method employs an algorithmic and unbiased approach that automates the creation of a systems integration plan that provides not only a roadmap for complexity reduction, but also cost estimates for milestone evaluation. Inspired by a recent large IS integration program, an example C-DSM of 100 interrelated legacy systems was created. The C-DSM method indicates that if a slow path to integration is selected then cost savings are estimated to surpass integration costs after several iterations.

15. SUBJECT TERMS

Systems integration, legacy information systems, collapsing design structure matrices

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	UU	142	Jeffrey Ogden, AFIT/ENS
					19b. TELEPHONE NUMBER (Include Area Code) (937) 255-6565, ext 4653 Jeffrey.ogden@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18